

# المدخل إلى لغة الفوُتران



د. سالم أحمد سحاب  
د. محمد يحيى عبد الرحمن

الناشر  
مركز النشر العلمي  
جامعة الملك عبد العزيز  
جدة ١٤٠٦ هـ (١٩٨٦ م)



# المدخل إلى لغة الفوريان

تأليف

د. محمد يحيى عبد الرحمن

أستاذ مساعد - قسم علوم الحاسبات  
كلية العلوم - جامعة الملك عبد العزيز

د. سالم أحمد محمد

أستاذ مساعد - قسم الرياضيات  
كلية العلوم - جامعة الملك عبد العزيز

الناشر

مركز النشر العالمي جامعة الملك عبد العزيز

ص ب ١٥٤٠٠ جدة ٢١٤٤١

المملكة العربية السعودية



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

يَرْفَعُ اللَّهُ الَّذِينَ آمَنُوا مِنْكُمْ وَالَّذِينَ  
أُوتُوا الْعِلْمَ دَرَجَاتٍ .

”صَدَقَ اللَّهُ الْعَظِيمُ“





© ١٤٠٦ هـ (١٩٨٦ م) جامعة الملك عبد العزيز .

جميع حقوق طبع هذا البحث محفوظة وملك الجامعة . مسموح بتخزينه في أي بنك للمعلومات والاقتباس منه دون إذن من صاحب الحق . غير مسموح بطبعه كاملاً ، أو نقله على أية هيئة أو بأية وسيلة ، سواء كانت إلكترونية ، أو شرائط مغنطية ، أو ميكانيكية ، أو استنساخاً ، أو تسجيلاً ، أو غير ذلك من الوسائل إلا بإذن كتابي من صاحب حق الطبع .

الطبعة الأولى ١٤٠٦ هـ (١٩٨٦ م)

إهداء

إلى والدينا :

عليهما من الله سبحانه الرحمة  
والتمننا بالغيفران .

المؤلفان

بسم الله الرحمن الرحيم

الحمد لله والصلاة والسلام على سيدنا محمد وعلى آله وصحبه ومن سار على هديه الى يوم الدين ،  
وبعد :

فإن الحاجة ماسة جداً لتأليف كتاب باللغة العربية يغطي احتياجات معظم طلاب الجامعات العربية وطلاب الكليات العلمية خاصة في مجال دراسة البرمجة باستعمال الفورتران «FORTRAN» والتي تعد من أهم اللغات المتداولة في مجال البرمجة على الحاسب الآلي بل أولها وأهمها على الإطلاق من حيث وفرها العلمية وإمكانياتها التطبيقية في شتى مجالات العلوم .

ومن هذا المنطلق عزمنا مستعينين بالله متكلين عليه على تأليف هذا الكتاب وتقديمه كأحد الكتب الممكن أن تؤدي للطلاب أن شاء الله الغرض المنشود من دراسة مبدئية في علم الحاسبات وتزوده بالمعلومات الأساسية والضرورية لتغطية منهج عن لغة الفورتران وبعض تطبيقاتها العلمية في مدة زمنية تعادل فصلاً دراسياً واحداً . كما أن الكتاب سيساعد على إعطاء الطالب المتخصص القاعدة العريضة الصلبة التي يحتاجها في مجال التخصص في علوم الحاسبات الآلية إن شاء الله .

وقد روعي في هذا الكتاب أن لا يكون له متطلب سابق على مستوى الجامعة ويعنى آخر فلن كل ما يحتاجه الطالب أو الدارس لهذا الكتاب هو الإلمام بمبادئ علم الجبر أو مبادئ الرياضيات الحديثة التي تدرس غالباً في المدارس الثانوية .

أما أسلوب الكتاب فهو يعتمد على محاولة التوفيق بين استيعاب الطالب للمادة العلمية النظرية وبين زيادة قدرته على البرمجة الفعلية وتطبيق الناحية النظرية في أسلوب متدرج بسيط ينمي قدرات الطالب ويوجد الحافز المشجع الذي يدفع الطالب قدماً لقضاء وقت أطول بين جنبات المعمل عن طواعية ورغبة في استيعاب المزيد عن إمكانيات اللغة وتطبيقاتها الواسعة فضلاً عن الرغبة في انجاز ما يوكل اليه من كتابة وبرمجة بعض المشكلات والمسائل المطلوبة منه .

وفي الختام نسأل الله العليّ القدير أن يجعل هذا العمل المتواضع البسيط لبنة بناءة من لبنات المكتبة العربية العلمية التي تساهم في تطور الأمتين العربية والإسلامية في مجالات العلوم الحديثة والله من وراء القصد .

المؤلفان



## محتويات الكتاب

رقم الصفحة

١	مقدمة	
	الفصل الأول : تطور الحاسبات والبرمجة .	

١-١	تطور الحاسبات الألكترونية العديدة	١
٢-١	الوحدات الأساسية المكونة للحاسبات العديدة	٣
٣-١	البيانات اللازمة لحل مشكلة ما على الحاسبات الألكترونية	١٢
٤-١	المراحل المختلفة التي يمر بها برنامج على الحاسبات الألكترونية	١٦

## الفصل الثاني : الثوابت والمتغيرات والعمليات الحسابية .

١٩	مقدمة	
١-٢	التخطيط لكتابة برنامج	٢١
٢-٢	بعض العمليات الأساسية للحاسب الآلي	٢٦
٣-٢	برنامج حساب الاستحقاق السنوي	٢٧
٤-٢	طريقة كتابة برنامج ما	٢٩
٥-٢	تسمية المتغيرات	٣١
٦-٢	قيمة المتغير	٣٢
٧-٢	أنواع قيم المتغيرات	٣٣
٨-٢	الحملة المبنية ( التوضيحية )	٣٤
٩-٢	موقع الجملة المبنية	٣٧
١٠-٢	التخصص التلقائي	٣٨
١١-٢	جملة التعيين	٣٩
١٢-٢	ترتيب العمليات الحسابية	٤٠
١٣-٢	ضرورة استعمال الأقواس	٤٣
١٤-٢	إشارة المساواة	٤٣
٤٦	« تمارين عامة »	

## الفصل الثالث : تعليمات إدخال وإخراج البيانات .

٤٩	مقدمة	
١-٣	جملة القراءة	٥٠
٢-٣	وصف البيانات	٥١
٣-٣	جملة الطباعة	٥٤

## ط

٥٧	استخدام الحرف A في الجملة الشارحة	٤-٣
٥٩	استخدام الحرف E في الجملة الشارحة	٥-٣
٦٢	استخدام الحرف H في الجملة الشارحة	٦-٣
٦٤	استخدام الحرف X في الجملة الشارحة	٧-٣
٦٥	استخدام علامة القسمة في الجملة الشارحة	٨-٣

### الفصل الرابع : تعليمات التحكم .

٦٧	مقدمة	
٧١	تعليمات التحكم .. إذا IF	١-٤
٧١	تعليمات إذا الحسابية	١-١-٤
٧٤	تعليمات إذا المنطقية	٢-١-٤
٧٧	ملحوظات على تعليمات التحكم إذا .. IF	٣-١-٤

### الفصل الخامس : تعليمات الانتقال .

٧٩	مقدمة	
٧٩	تعليمات الانتقال GO TO الغير مشروطة	١-٥
٨٣	تعليمات الانتقال GO TO المشروطة	٢-٥
٩٠	ملاحظات على تعليمات الانتقال GO TO ، إذا IF	

### الفصل السادس : تعليمات حلقة التنفيذ المتكرر .

٩٦	مقدمة	
١٠٠	الصورة العامة لتعليمات حلقة التنفيذ المتكرر	١-٦
١٠٥	تعليمات الاستمرار أو للمواصلة	٢-٦
١٠٩	حلقات التنفيذ المتداخلة	٣-٦
١١٣	ملحوظات على استخدام حلقة التنفيذ المتكرر	٤-٦
١١٨	ملاحظات على استخدام حلقات التنفيذ المتكرر	٥-٦
١٢٤	« محارين »	

### الفصل السابع : المصفوفات .

١٢٧	مقدمة	
١٢٧	أهمية المصفوفات	١-٧
١٣١	المصفوفات	٢-٧
١٣٢	الجملة المهيئة للمصفوفة	٣-٧
١٣٤	عناصر المصفوفة	٤-٧

١٣٥	جملة تحديد البعد .....	٥-٧
١٤٠	المصفوفات وحلقات التنفيذ .....	٦-٧
١٤١	بعض الأخطاء الشائعة عن المصفوفات .....	٧-٧
١٤٢	المصفوفة ذات الدليلين العدديين .....	٨-٧
١٤٤	ادخال واخراج بيانات المصفوفات .....	٩-٧
١٤٧	تداخل الطرق المتكررة المباشرة .....	١٠-٧
١٥٦	« تمارين عامة » .....	

## الفصل الثامن : البرامج الجزئية .

١٦١	مقدمة .....	
١٦٢	الدوال سابقة التخزين .....	١-٨
١٦٦	الدوال ذات التصير الرياضي .....	٢-٨
١٦٧	الدوال في صورة برامج جزئية .....	٣-٨
١٧٣	البرامج الجزئية الفرعية .....	٤-٨
١٧٥	تعليمة استدعاء البرنامج الجزئي الفرعي .....	١-٤-٨
١٨٠	الفروق بين البرامج الجزئية للذالة والبرامج الجزئية الفرعية .....	٢-٤-٨
١٨١	تعليمة التعميم .....	٥-٨
١٨٧	تعليمة التكافؤ .....	٦-٨
١٨٨	الصورة العامة لتعليمة التكافؤ .....	١-٦-٨
١٩١	الارتباط بين تعليمتي التعميم والتكافؤ .....	٢-٦-٨
١٩٣	« تمارين » .....	
١٩٥	تمارين عامة .....	
٢٠٨	قاموس المصطلحات .....	
٢١٧	مراجع .....	





## الفصل الأول تطور الحاسبات والبرمجة



## الفصل الأول

### Electronic Digital Computers

### ١-١ تطور الحاسبات الألكترونية العددية

لن ندهش كثيراً اذا علمنا أن عصر الحاسبات قد بدأ التفكير فيه في سنوات ما قبل الميلاد ( من سنة ٤٠٠٠ الى سنة ٣٠٠٠ قبل الميلاد ) ، حيث بدأ التفكير فيها باستخدام مستطيل معدني مركب عليه خيوط أو أسلاك متوازية وفي كل منها عدد من الكرات الصغيرة ذات الألوان المختلفة حيث كل لون منها يمثل قوة حسابية معينة ( آحاد - عشرات ... الخ ) ، وقد اطلق على هذه الآلات اسم الحاسبات calculi .

وفي منتصف القرن السادس عشر امكن تصميم أولي الحاسبات الميكانيكية والتي تعتمد في إجراء عمليات الجمع والطرح على مجموعة من التروس والبكرات ، وأمكن تحسينها بعد ذلك بحيث يمكنها إجراء عمليات الضرب بطريقة مباشرة بعد أن كانت تتم بأجراء عمليات جمع متتالية .

وفي حوالي سنة ١٨٣٠ أمكن للرياضي البريطاني شارل بباچ Charles Babbage أن يضع تصوراً لآلة حاسبة تحتوي على :

وسيلة لإدخال بيانات - امكانية تخزين - امكانيات عمل بعض الحسابات - وسيلة لإخراج نتائج . وقد اطلق على آله اسم الآلة التحليلية The analytical engine .

وبعد حوالي مائة عام أمكن تحقيق تصورات ذلك العالم البريطاني عندما بدى في تصميم وبناء الحاسب الذي سمي MARK I ، وكان ذلك في جامعة هارفارد عام ١٩٣٩م وفي عام ١٩٤٤م بدى في تشغيل ذلك الحاسب وأعتبر هذا العام كفتحة لعصر الحاسبات . ومع أن هذا الحاسب كان في البداية يعتمد في تصميمه على دوائر كهربائية ميكانيكية إلا أنه أمكن بعد ذلك تطويره بإدخال الصمامات في دوائر الكهربية مما جعله أسرع وأطلق عليه حينذاك اسم ENIAC وكان ذلك أول حاسب الكتروني رقمي (Digital) . وكانت العقبة التي واجهت مثل تلك الحاسبات في ذلك الوقت هو تخزين التعليمات والبيانات اللازمة لكل مشكلة يراد حلها ، الى أن أمكن التغلب على تلك العقبة بعد سنوات قليلة باستخدام نظام البطاقات المثقوبة والذي كان قد ابتكره الاحصائي هيرمان هيوليتر Herman Hollerith عام ١٨٨٩م - وقد اطلق على أول نوع من تلك الحاسبات التي تحتوي دوائر تخزين بيانات ومعلومات بأسم UNIVAC ، وظلت الصمامات الكهربية هي المكون الأساسي للحاسبات حتى أواخر عام ١٩٥٠م حيث ظهرت المقومات والتي تعرف بأسم الترانزستور والتي كانت البديل الأمثل للصمامات سواء في جعل الحاسبات أقل حجماً وتكلفة وأقل انبعاثاً

للحرارة التي كانت تحدثها الصمامات والتي بالتالي كانت تتطلب تكلفة أكثر في تجهيزاته . وقد اطلق على الفترة من ١٩٥٠م الى ١٩٦٥م بأسم فترة الجيل الثاني من الحاسبات .

وقد أمكن بعد ذلك تطوير هذا الجيل من الحاسبات بأدخال نظام الدوائر المتكاملة *Integrated Circuits* والذي ساعد كثيراً في تقليل حجم الحاسبات وتكلفتها وزيادة سرعة كفاءتها في اجراء العمليات الحاسوبية كما أمكن ادخال تعديلات كبيرة في تصميماتها الداخلية والخارجية . وقد كانت الحاسبات من طراز أ. ب. م *IBM 360, IBM 370* هي رائدة هذا الجيل من الحاسبات والذي أطلق عليه الجيل الثالث لتطور الحاسبات .

وفي وقتنا الحاضر والذي يعتبر الجيل الرابع للحاسبات ادخلت كثير من التعديلات في تصميمات الحاسبات وذلك بفضل التقدم الذي يحدث حالياً في مجال تصنيع الدوائر المتكاملة والذي أدى الى امكانية انقاص حجم الحاسبات وتقليل تكلفتها وزيادة حجم التعليمات والبيانات التي يمكن تخزينها وزيادة سرعة اجراء العمليات الحاسوبية والمنطقية التي يقوم بها .

ولا شك أننا نلمس الآن مدى ما أحدثته الحاسبات من تقدم في جميع المجالات سواء الحربية ( العسكرية ) أو المدنية حتى أصبحت العصب الرئيسي لكثير من الأنشطة والقطاعات لأية دولة متقدمة ، نخذ مثلاً الأسلحة العسكرية المتطورة فالصواريخ والغواصات والطائرات الحربية والأقمار الصناعية وحتى الدبابات الحديثة وغيرها من الأسلحة تعتمد على الحاسب الإلكتروني الى درجة عالية جداً بحيث أنها تفقد فعاليتها اذا أختل عمل الحاسب الإلكتروني لسبب أو لآخر ، بل أن بعض طائرات التجسس الحديثة تعتمد اعتماداً كلياً على الحاسب الإلكتروني في أداء مهمتها التصويرية والتجسسية .

أما من الناحية المدنية فلا يكاد يخلو قطاع واحد من قطاعات الدولة إلا وهو ماض قدماً في سبيل إدخال الحاسب الإلكتروني لتسهيل عمل ذلك القطاع ومن أهم هذه القطاعات وزارة الداخلية الذي من خططه الحالية تخزين المعلومات الأساسية عن جميع أفراد الشعب السعودي بواسطة الحاسب الإلكتروني لتحل محل استعمال الملفات التقليدية التي تستهلك الكثير من الجهد والوقت بالإضافة الى إمكانية تلفها أو فقدانها وإستخدامها لمساحات كبيرة لحفظها .

ومن الأمثلة الأخرى إعتاد شركات الطيران الحديثة ومنها السعودية على الحاسب الإلكتروني الذي يقوم بأعطاء جميع المعلومات الضرورية عن الحجز وأرقام الرحلات ومواعيد الاقلاع والوصول ... الخ ، والأمثلة كثيرة وعديدة في المجالات العلمية المختلفة كالطب والهندسة وعلوم الفضاء .

## ١-٢ الوحدات الأساسية المكونة للحاسبات العددية :

لنفرض أننا نعيش الفترة التي سبقت ظهور الحاسبات وطلب منا حل إحدى المشكلات الرياضية العددية ، وليس لدينا من أدوات لحل تلك المشكلة سوى الورقة والقلم . فلنبدأ الحل لاهد من توافر البيانات الخاصة بالمشكلة وتطبيق تلك البيانات على طريقة الحل سيمكنا الحصول على النتائج المطلوبة .

فعل سبيل المثال ، لنفرض أننا نريد حساب  $Y$  والتي تعطي من العلاقتين .

$$X = A + B - C, Y = \frac{X}{D}$$

فلحساب قيمة  $Y$  فإن ذلك يتطلب عدة تساؤلات أساسية منها :-

- ١ ) ماهي المتغيرات التي لاهد من معرفة قيمتها كي تؤدي في النهاية الى معرفة قيمة  $Y$  ؟ .
  - ٢ ) بأي وسيلة سيم إجراء العمليات الحسابية والى أي درجة من الدقة ؟ .
  - ٣ ) ماهو تسلسل إجراء العمليات لحل تلك المشكلة ؟ .
  - ٤ ) هل هناك وسيلة لحفظ النتائج الوسيطة والتي سيتطلب استخدامها في مراحل لاحقة للحصول في النهاية على الحل ؟ .
  - ٥ ) ماهي النتائج المطلوب الحصول عليها لمثل حلا للمشكلة ؟ .
- ولنبداً بالأجابة عن التساؤلين الأول والثالث . فمن العلاقتين السابقتين يتضح بوضوح عدم إمكانية حساب قيمة  $Y$  قبل حساب قيمة  $X$  . ولكن لحساب قيمة  $X$  ، فلا بد من معرفة قيم  $A, B, C$  وكذلك لحساب قيمة  $Y$  فيجب أيضاً معرفة قيمة  $D$  .

أي أننا لحساب قيمة  $Y$  لاهد أولاً من معرفة قيم المتغيرات  $A, B, C, D$  وهذا ما يجيب على التساؤل الأول . وما سبق يتبين أيضاً أن تسلسل إجراء العمليات سيكون كالآتي :

١ ) معرفة قيم  $A, B, C, D$

٢ ) حساب قيمة  $X$  من العلاقة :  $X = A + B - C$

٣ ) حساب قيمة  $Y$  من العلاقة :

$$Y = \frac{X}{D}$$

٤ ) كتابة قيمة  $Y$  المطلوب الحصول عليها .

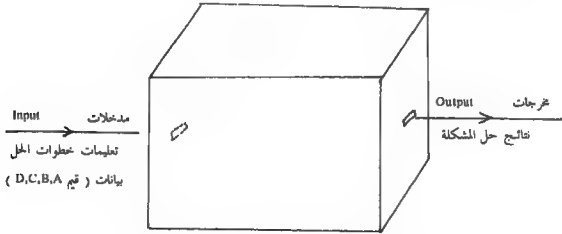
كذلك يتضح أن تلك المشكلة تتطلب اجراء بعض العمليات الحسابية الكلاسيكية مثل الجمع والطرح والقسمة ، ولحسن الحظ أنها لا تحتوي على عمليات حسابية خاصة مثل إيجاد جذر تربيعي

أو لو غاريم عدد ما أو حساب قيم مرفوعة الى قوى أكبر أو أصغر من الواحد أو دوال مثلثية أو غير ذلك . ولذا فقد يتطلب الأمر إجراء مثل تلك الحسابات في مشكلة أخرى ، وبالتالي فلا بد من تواجد وسيلة تستخدم لعمل تلك الحسابات سواء كانت تلك الوسيلة يدوية أو آلات بدائية . وفي الحالتين لابد من تدخل العنصر البشري وما يستتبع ذلك من تفاوت في السرعة والدقة ، ولعل ذلك يجيب على التساؤل الثاني .

وحيث أننا افترضنا أن الأدوات التي لدينا لا تتمدى الورقة والقلم ، فإنه من البديهي أن تكون الورقة وما عليها من حسابات هي الوسيلة الوحيدة لحفظ ( تخزين ) البيانات والنتائج الوسيطة ( قيمة  $X$  وقيم المتغيرات  $A, B, C, D$  في مثالنا الحالي ) . وفي هذه الحالة فإن سرعة الحصول على عمليات حسابية وسيطة سبق تلويئها ويراد استخدامها فيما بعد ، ستتوقف على العنصر البشري أيضاً وطريقة تنظيمه في وضع تلك الحسابات الوسيطة على سطح الورقة .

وأخيراً لابد لنا أن نتساءل ماهو الهدف الذي نريد أن نصل اليه بعد إجراء العمليات الحسابية الوسيطة وتجميعها والتعويض بها في العلاقتين السابقتين ؟ والاجابة بديهية ، وهي الحصول على قيمة  $Y$  بلوحة تقريب تتوقف على العمليات الحسابية الوسيطة والنهائية . ولذا فإن العمليات الحسابية الوسيطة ( حساب قيمة  $X$  ) لم تكن الا مرحلة وجب لحفظها والاحتفاظ بها لاستخدامها في الوصول الى الهدف النهائي وهو حساب قيمة  $Y$  .

والآن لتتخيل مرة أخرى جهازاً على شكل صندوق كبير لا نعرف ماذا يحوي بداخله ولا يظهر منه سوى فتحتين متقابلتين الشكل (١-١) . ومتتخيل أن إحدى الفتحتين ستكون مخصصة لادخال قيم  $A, B, C, D$  بطريقة ما ، وكذلك خطوات حل المشكلة .



شكل (١-١)

والفتحة الأخرى للحصول على نتيجة قيمة Y بطريقة ما أيضاً . أي على النتيجة النهائية للمشكلة ، فماذا يمكن أن نتوقع وجوده بداخل ذلك الصندوق ؟

- وللاجابة على ذلك التساؤل سنجد أن هذا الصندوق لابد وأن تكون له القدرة على :
- ( ١ ) ترجمة وتنفيذ خطوات حل المشكلة بالترتيب الذي أدخلت له ،
  - ( ٢ ) تخزين بيانات المشكلة وخطوات حلها وكذلك النتائج الوسيطة التي تلزم للحصول على النتائج النهائية .
  - ( ٣ ) القيام بتنفيذ العمليات الحسابية التي تتطلبها حل المشكلة .
  - وفي الواقع إذا أضفنا للثلاث قدرات السابقة إمكانية :
  - ( ٤ ) ادخال تعليمات خطوات الحل والبيانات الخاصة بالمشكلة .
  - ( ٥ ) إخراج النتائج النهائية والوسيطة ( ان تطلب الأمر ذلك ) .

لحصلنا على حاسب عددي تتوقف قدراته على مدى كفاية الحس وحدات أو قدرات السابق ذكرها . وبما سبق يمكننا أن نستخلص أن الوحدات الأساسية المكونة لأي حاسب عددي هي :

#### ١ ) وحدة لأدخال التعليمات والبيانات Input Data Unit

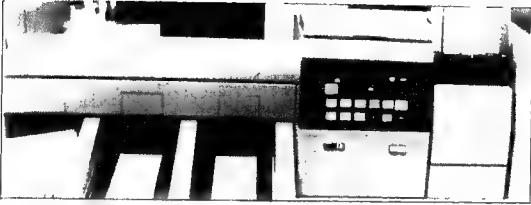
وتكون مهمتها قراءة التعليمات والبيانات . ونقصد هنا بالتعليمات ، خطوات الحل مرتبة في تسلسل منطقي في صورة أوامر أو تعليمات Statements or Instructions .

أما البيانات Data فتقصد بها القيم العددية التي تلزم لحل المشكلة . ومن هذا يتضح أن التعليمات والبيانات ستكون مزيجاً من الحروف Letters والأعداد Numbers والعلامات الخاصة Special characters ( والتي سنذكرها فيما بعد ) .

وتوجد في وقتنا الحاضر وسائل عديدة يمكن عن طريقها ادخال التعليمات والبيانات الى الحاسبات العددية منها :

#### ( أ ) وحدة قراءة البطاقات المثقبة Card Reader Unit

وكانت تعتبر حتى أواسر الستينات من أهم الطرق وأكثرها شيوعاً . وفي هذه الطريقة يتم وضع التعليمات والبيانات على بطاقة ورقية ذات سُمك وملمس خاص وفي صورة بحيث أن كل رقم أو حرف أو علامة خاصة تمثل بثقب أو مجموعة رأسية من الثقوب تميز ذلك الحرف أو الرقم أو العلامة عن غيره من الحروف والأرقام والعلامات . ويتم قراءة تلك البطاقات المثقبة عن طريق معرفة الثقوب أو مجموعة الثقوب الموجودة في المواضع المختلفة من البطاقات وترجمتها الى نبضات كهربائية Electric pulses تمثل الحرف أو الرقم الذي تم قراءته ، ( الشكل ١-٢ ) .



شكل (٢-١)

وقد بدأ الاعتماد على تلك الطريقة يقل تدريجياً بظهور طرق تكنولوجية أخرى حديثة مثل وحدات القراءة عن طريق الأشرطة المغناطيسية أو الاسطوانات أو الأقراص المغنطة وجميع تلك الطرق الحديثة تمتاز في سرعة ادخال البيانات للحاسب وفي حجم البيانات التي يمكن أن تستوعبها على مساحات صغيرة منها .

**(ب) وحدة قراءة الشرائط الورقية المثقبة : Punched Paper tape reader unit:**

وتشبه الوحدات السابقة ، إلا أن التعليمات والبيانات يتم تثقيبها على شريط ورقي ، كما أن طريقة قراءة تلك الثقوب تشبه تلك المستخدمة في البطاقات المثقبة .

**(ج) وحدة قراءة الشرائط والأقراص المغنطة :**

**Magnetic tape or disc reader unit:**

ويتم تسجيل البيانات والتعليمات على أشرطة أو اسطوانات ممغنطة شبيهة بتلك التي تستخدم في أجهزة المسجلات Tape Recorders ثم تغذى بها وحدة القراءة .

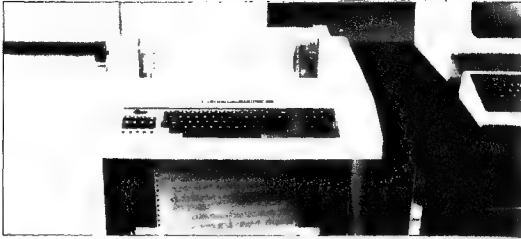
**(د) وحدة ادخال البيانات والتعليمات عن طريق الوحدات الطرفية :**

**The Teletype devices:**

ويعتبر إحدى الوسائل الشائعة الاستخدام في ادخال البيانات في الحاسبات وتمتاز بسهولة استخدامها وإمكانية تغذية الحاسبات بالبيانات والتعليمات عن طريقها من مسافات بعيدة ( الشكل ٣-١ ) .

ولذا فكثيراً ما تستخدم مثل هذه الوحدات في الأغراض التجارية مثل شركات الطيران والبنوك والمؤسسات الكبرى حيث ترتبط جميعها بالحاسب عن طريق توصيلات تليفونية وسبب سهولة استخدامها هو في أنها تشبه الآلات الكاتبة Typewriters في استخدامها كما أن البيانات والتعليمات





شكل (١-٣)

التي تعطي للحاسب عن طريقها تظهر مكتوبة واضحة بحيث تقلل من فرصة دخول أرقام أو حروف غير صحيحة للحاسب .

(هـ) وحدة ادخال البيانات عن طريق الوحدات الطرفية ذات الشاشة :

#### Cathode Ray Tube Screens or Video Devices:

وتعتبر هذه الوحدات استحداثا للوحدة السابقة د ، حيث أن البيانات والتعليمات التي يغذى بها الحاسب تظهر في نفس الوقت على شاشة تليفزيونية مضيئة .



شكل (١-٤)

وبمجموع هذه الوحدات توجد وسائل أخرى أقل استخداما وشيوعا مثل : الكاسيتات المغنطة Magnetic cassetts ، الحبر المغنط Magnetic Ink ، الاسطوانات المغنطة Magnetic discs ، البطاقات المغنطة Magnetic cards ، وكذلك عن طريق الاتصال الصوتي Voice communication . التي تستخدم في نطاق ضيق ومحدود .

### ٢ ) وحدة استخراج النتائج : Output Unit

.وتتركز مهمتها في إستخراج النتائج الوسيطة والنهائية ، بجانب كتابة البيانات والمعلومات التي سبق قراءتها عن طريق وحدة الإدخال . وأهم وحدات استخراج النتائج المعروفة في وقتنا الحاضر هي :

#### الطابعات : Line Printers

وتعتبر وسيلة الاستخراج الأكثر شيوعاً ولا يخلو منها أي حاسب عددي ، فمن طريقها يمكن الحصول على النتائج مطبوعة على الورق بحيث يمكن استخدامها مباشرة بخلاف بعض وحدات الاستخراج الأخرى والتي قد تتطلب أجهزة مساعدة لترجمة المستخرجات كي تظهر بالصورة التي يمكننا أن نقرأها بها ، كما يعتبر الطابع أحد الوسائل السريعة والسهلة الاستخدام .

ويمكن اعتبار وحدات الإدخال والتي سبق أن أشرنا إليها كوحدات استخراج أيضاً .

### ٣ ) وحدة التخزين : Storage or Memory Unit

وفها يتم تخزين جميع البيانات والتعليمات التي تعطى للحاسب عن طريق وحدات الإدخال وكذلك الأوامر الخاصة بحل المشكلة بترتيبها المنطقي ، وأخيراً نتائج العمليات الحسابية الخاصة بالمشكلة وبالتالي النتائج النهائية لها .

وتعتبر وحدات التخزين في الحاسبات إحدى عوامل تفضيل ما بين حاسب وآخر من ناحيتين :

( أ ) كمية البيانات والتعليمات التي يمكن أن تستوعبها والتي تعرف علمياً بسعة التخزين Storage capacity .

(ب) سرعة تجاوب وحدة التخزين والتي تقاس بماملين :

١ ) الزمن اللازم لتخزين معلومة .

٢ ) الزمن اللازم لقراءة تلك المعلومة أي الزمن الذي تأخذه وحدة التخزين في البحث عن موضع معلومة ما بداخلها .

ومجموع الزمنين السابقين هو ما يطلق عليه بأسم زمن التجاوب Access time وبالتالي فكلما كانت سعة وحدة التخزين لحاسب ما كبيرة وزادت سرعة تجاوبها ( أي كان زمن التجاوب بها قصيراً ) كلما أرتفعت كفاءة ذلك الحاسب وكان أفضل من غيره من الحاسبات .

ووحدات التخزين في الحاسبات الألكترونية الحديثة تتركب من حلقات ممغنطة Magnetic cores أو من دوائر متكاملة Integrated circuits وكل موضع في وحدة التخزين يمكن التعرف عليه

عن طريق رقم يرمز له ويطلق عليه اسم عنوان Address . ويجب هنا أن نفرق بين عنوان معلومة ما وبين ما يوجد من بيانات في داخل هذا العنوان . ففي المثال السابق إذا فرضنا أن قيمة A هي 7 ، B ، 28 هي C ، 2 هي 2 ، وكانت عناوين A ، B ، C هي على التوالي :

	00	01	02	03	04	05	06	07	08	09	10	11
00												
01												
07			7									
08												
09				28								
12			2									

شكل (٥-١)

فأنتنا نقول مثلاً أن بداخل الموضع الذي عنوانه 0701 ( وهو يمثل موضع A ) توجد القيمة 7 أي أن قيمة A هي 7 . ويقوم الحاسب بموضع عنوان لكل متغير يتم تخزينه ، مالم يعط أمراً لتخزين متغير ما في موضع معين بوحدة التخزين .

وبعض الحاسبات وخاصة الكبيرة منها تحتوي على أكثر من وحدة تخزين احدهما أساسية أو مركزية Main Storage or Central memory ولا بد من تواجدها في أي حاسب وتتميز بسرعة تجاوبها وسعتها المحدودة نسبياً . أما بقية وحدات التخزين ( ان وجدت ) فيطلق عليها وحدات تخزين ثانوية Auxiliary Secondary memory وهي تتميز بسعة تخزين كبيرة ولكن بسرعة تجاوب أقل نسبياً من الأولى .

#### ٤ ) وحدة الحساب والمنطق : The Arithmetic Logic Unit

وهي الوحدات التي تقوم بتنفيذ العمليات الحسابية العددية والمنطقية على البيانات التي سبق تخزينها في الحاسب وطبقاً لتسلسل تنفيذها في مجموعة التعليمات والأوامر والتي تكون ما يعرف باسم البرنامج The Program ( والذي ستطرق الى شرحه بالتفصيل في الباب الثاني ) . فلاجراء

عملية حسابية على متغيرين فإنه يتم تخزين قيمتي هذين المتغيرين في وحدات تسجيل Registers موجودة داخل تلك الوحدة والتي يتم في أحدها تسجيل نتيجة تلك العملية تمهيداً لتخزينها في المكان المخصص لها بوحدة التخزين الرئيسية للحاسب .

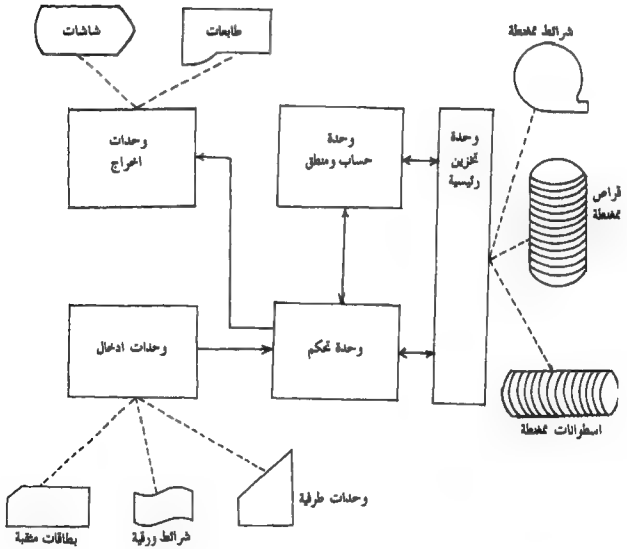
ويلاحظ من إسم تلك الوحدة أنها تقوم بتنفيذ العمليات الحسابية العددية مثل الجمع والطرح والضرب... الخ وكذلك العمليات المنطقية Logical Instructions مثل المقارنة بين قيمتين من ناحية الكبر أو الصغر أو التساوي أو عدم التساوي أو عمليات منطقية أخرى ستعرض لها في حينها .

وكما لوحدة التخزين من أهمية في المفاضلة بين حاسب وآخر فكل ذلك الوحدة الحسابية ، حيث تقاس كفاءة الحاسب بالنسبة لهذه الوحدة بمدى تجاوب وسرعة تنفيذ العمليات الحسابية والمنطقية . وقد نلمس مدى الفارق في التطور الذي حدث في تلك الوحدة بالذات اذا تذكرنا حاسبات الجيلين الأول والثاني والتي كانت تعتمد في تنفيذ العمليات الحسابية على تمشيق تروس وادارتها وما يتطلبه ذلك من زمن ومجهود أو في استخدام الصمامات وما كان يتطلبه ذلك من تجهيزات أخرى في الغلب على الحرارة الكبيرة المنبعثة منها والرقابة والصيانة المستمرتين على كفاءة عمل تلك الصمامات بجانب ارتفاع أثمانها بالمقارنة باستخدام الدوائر المتكاملة أو الترانزستورات المستخدمة حالياً في تركيب تلك الوحدات .

#### ٥ ( وحدة الرقابة أو التحكم : The Control Unit

وتعتبر هذه الوحدة من أنشط الوحدات السابقة فهي في عمل دائم ومستمر طالما أن الحاسب يقوم بتنفيذ عملية حسابية ما ، اذ تشتمل هذه الوحدة على الدوائر التي تتحكم في وضع البيانات والتعليمات داخل وحدة التخزين الرئيسية للحاسب وذلك بترتيبها وترجمة كل منها ، وكذلك التعامل مع الوحدة الحسابية والوحدات الأخرى لاجراء وتنفيذ تلك التعليمات . وتقوم وحدة التحكم بتخزين التعليمات والأوامر التي يحتويها البرنامج في أماكن محددة من وحدة التخزين ( معروفة عناوينها ) كما تقوم بتخزين البيانات الخاصة بالمشكلة في أماكن أخرى ثم تبدأ وحدة التحكم في تنفيذ تلك التعليمات حسب تسلسلها في البرنامج الذي يراد الحساب عليه وعند تنفيذ تعليمة ما تقوم وحدة التحكم بالتعرف على نوع العملية المراد تنفيذها ( جمع أم طرح... الخ ) وذلك بمعرفة شفرة العملية Operation Code والبيانات التي تلزم لتنفيذها ونقل تلك الشفرة والبيانات الى وحدات التسجيل في وحدة الحساب والمنطق لتنفيذها ثم تقوم تلك الوحدة بنقل النتيجة الى المكان المخصص لها في وحدة التخزين .

ولذا فإنه يتبين مما سبق أن هناك ترابطاً تبادلياً بين وحدة التحكم وكل من وحدة الحساب والمنطق ووحدة التخزين . وعموماً فإنه يمكن تمثيل الترابط بين الوحدات المختلفة السابقة وبين بعضها وكذلك وحدات الإدخال والإخراج ووحدات التخزين الثانوية كما هو مبين في ( الشكل ٦-١ ) .



شكل (٦-١)

### ١-٣ البيانات اللازمة لحل مشكلة ما على الحاسبات الإلكترونية :

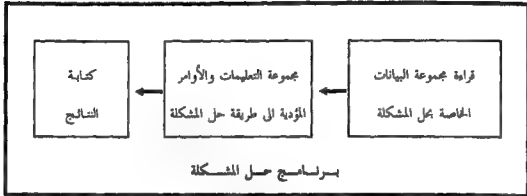
عند بدء ظهور الحاسبات الإلكترونية في العالم العربي أطلق البعض عليها اسم « العقول الألكترونية » ولا شك أن هناك farkاً كبيراً بين الحاسب والعقل . فبدون أن يمهّد العقل للحاسب طريقة الحساب والبيانات اللازمة للمشكلة فلن يستطيع الحاسب أن ينفذ عملية حسابية واحدة ولو كانت من أبسطها . وكل ما يمتاز به الحاسب عن العقل البشري هو سرعة إنجاز العمليات الحسابية والدقة في حسابها ، ففي الزمن الذي يستغرقه العقل البشري في عملية جمع عددين بدرجة تقريب معينة تستطيع بعض الحاسبات تنفيذ أكثر من مليون عملية جمع لأعداد تحتوي على ٣٢ أو ٦٠ رقماً .

وسبق أن عرفنا أن هناك وحدتين من الوحدات الأساسية للحاسبات الألكترونية هما وحدتي ادخال البيانات والتعليمات - ووحدة اخراج النتائج . ولذا فلا بد لنا أن نجهز أنفسنا بتحصير ما سيم اعطاه الحاسب من بيانات وتعليمات عن طريق وحدة الادخال ، وكذلك لا بد لنا من التفكير فيما نريد إستخراجه من نتائج للمشكلة ، هل نكتفي باستخراج النتائج النهائية للمشكلة ؟ أم يفضل استخراج نتائج وسيطة ، خاصة للعمليات الحسابية التي تتطلب وقتاً كبيراً وذلك للتأكد من صحة تلك النتائج النهائية .

وإذا تأملنا في أي مشكلة حسابية نجد أنها لا بد من أن تمر بمراحل ثلاث لحلها :

- ١ ( توفر البيانات اللازمة لحل المشكلة .
  - ٢ ( طريقة الحل (الخوارزم)<sup>(٥)</sup> The Algorithm . أو مجموعة التعليمات والأوامر التي لا بد للحاسب من تنفيذها .
  - ٣ ( تحديد النتائج المطلوب إستخراجها .
- وهذه المراحل الثلاث مجمعة تكون فيما بينها اسم البرنامج The Program وهي موضحة في ( الشكل ١-٧ ) .

(٥) الخوارزم : كلمة غربية تطلق على مجموعة خطوات تنفذ بطريقة تسلسلية بهدف الحصول على الحل النهائي . وهذه الكلمة أطلقها الغربيون إعترافاً وتقديراً للعالم المسلم محمد ابن موسى الخوارزمي الذي عاش في بغداد بين عام ١٦٤هـ وعام ٢٣٥هـ وتوفي هناك . وبرز في علم الرياضيات والفلك في عهد الخليفة المأمون . كما جلد أيضاً باحتكاره علم حساب «اللوغاريتمات» .



شكل (١-٧)

مثال (١) :

باستخدام الحاسب ، أوجد جذري معادلة الدرجة الثانية :  $AX^2 + BX + C = 0$

أولاً : يتطلب لابتعاد جذري المعادلة السابقة معرفة قيم  $A, B, C$  والتي يمكن اعتبارها أنها البيانات التي يجب توفرها .

ثانياً : اذا علمنا أن جذري معادلة الدرجة الثانية يعطى من القانونين :

$$X_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A} ; \quad X_2 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

فلا يتبقى الا أن نعطي الحاسب مجموعة التعليمات والأوامر بحساب قيمتي  $X_1, X_2$  . وطريقة الحل قد تختلف بين شخص وآخر ، فقد تكون مجموعة تعليمات وأوامر بالتعميض المباشر بقيم  $B, C, A$  في القانونين السابقين بغض النظر عما اذا كانت قيمة  $A$  تساوي صفراً أم لا ، أو أن تكون قيمة  $(B^2 - 4AC)$  سالبة ، والذي قد يتسبب عنها توقف الحاسب عن تنفيذ بقية البرنامج أو إعطاء نتائج مضللة .. ولهذا فإنه يجب علينا مراعاة الحالات التالية والتي سيترتب عليها توقف البرنامج في حالة حدوث أحدها كي نقوم بمعالجتها من خلال التعليمات المناسبة .

الحالة الأولى :

عندما تكون قيمة  $A$  تساوي صفراً فإن المعادلة تتحول الى معادلة من الدرجة الأولى  $BX + C = 0$  ويكون هناك جذر واحد هو  $X = -C/B$  .

## الحالة الثانية :

عندما تكون  $(B^2 - 4AC)$  سالبة ، ستكون الجذور مركبة وفي هذه الحالة يجب أن يحتوي الخوارزم على طريقة لحسابها .

## الحالة الثالثة :

تكون  $(B^2 - 4AC)$  غير سالبة وقيمة  $A$  تختلف عن الصفر عندئذ لن تكون هناك مشاكل في حساب قيمتي  $X_1$  ،  $X_2$  وكتابة قيمهما بعد أن يقوم الحاسب بالتعويض في القانونين السابقين .

فإذا اردنا المفاضلة بين مجموعتي التعليمات والأوامر سنجد أن الأخيرة تفوق الأولى في أنها تحوي جميع الاحتمالات التي قد يصادفها الحاسب عند تنفيذ البرنامج وما يجب عليه عمله عند ظهور أي منها ، لأننا كما ذكرنا أن الحاسب عديم القدرة على التفكير ولكنه سريع التنفيذ طالما أن التعليمات والأوامر واضحة ومحددة له .

## مثال (٢) :

بأستخدام الحاسب ، أوجد قيمة كثيرة الحدود المعطاه بالمعادلة :

$$Y = 2X^3 + 3X^2 + 5X + 3 \quad (1)$$

في هذه الحالة سنجد أنه لحساب قيمة كثيرة الحدود ، أي لإيجاد قيمة  $Y$  :

أولاً : يتطلب معرفة قيمة  $X$  ، حيث أنه عند كل قيمة للمتغير  $X$  سنجد أن  $Y$  ستكون لها قيمة . وإذا كانت كثيرة الحدود السابقة في الصورة :

$$Y = AX^3 + BX^2 + CX + D \quad (2)$$

فأنه في هذه الحالة يلزمنا معرفة قيم  $A$  ،  $B$  ،  $C$  ،  $D$  بجانب قيمة  $X$  حتى يمكننا في النهاية حساب قيمة المتغير  $Y$  . وسنفترض في مثالنا الحالي كثيرة الحدود المعطاه بالمعادلة (١) والتي يلزم لحسابها معرفة قيمة  $X$  فقط والتي يجب اعطاؤها للحاسب اذا أريد معرفة قيمة  $Y$  عند القيمة المعطاه للمتغير  $X$  .

ثانياً : اذا أعطينا الحاسب قيمة  $X$  فإن حساب قيمة  $Y$  يمكن الحصول عليها بأحدى طريقتين :  
(أ) بأعطاء الحاسب مجموعة التعليمات والأوامر بحيث يقوم بحساب  $5X$  ،  $2X^2$  ،  $3X^3$  ويقوم بجمعها ويضيف الى الناتج القيمة 3 . وفي النهاية نعطيه الأمر بكتابة قيمة  $Y$  .

(ب) بتطوير كثيرة الحدود أولاً وجعلها في الصورة :

$$Y = ((2X + 3)X + 5)X + 3 \quad (3)$$

وهي نفس المعادلة (١) جبرياً بعد فك الأقواس .



فإذا أردنا المقارنة بين الطريقتين سنجد أن الطريقة الثانية (ب) ستكون أسرع وأكثر دقة حيث أنها لن تتطلب من الحاسب أكثر من عمليات ضرب وجمع بينما الطريقة الأولى (أ) مستلزم حساب قوى مختلفة للمتغير  $X$  والتي ستطلب من الحاسب زمناً أطول (٥) وستكون النتيجة أقل دقة من عمليات الضرب فقط .

**الافتاء** من الواضح أن النتائج المطلوب استخراجها هي قيم  $Y$  عند إعطاء قيم  $X$  ، ولنا فقد يكون من الأنسب في هذا المثال إعطاء الحاسب الأمر بكتابة قيمة  $X$  المعطاه وقيمة  $Y$  المناظر لها .

ونستخلص مما سبق أنه لحل مشكلة ما على الحاسبات الألكترونية فإنه يلزم :

١ - إعطاء الحاسب مجموعة التعليمات والأوامر التي تمثل برنامج حل المشكلة والذي يحتوي على :

- ( أ ) أوامر قراءة البيانات اللازمة لحل المشكلة .
- ( ب ) طريقة حل المشكلة ، وإذا كان هناك أكثر من طريقة فإن درجة التفضيل تكون لتلك الأكثر دقة والأقل زمناً (إن أمكن) في الحصول على النتيجة .
- ( ج ) أوامر لكتابة النتائج سواء النهائية أم الوسيطة كذلك .
- ٢ - أن يشمل البرنامج - قدر الأمكان - على مختلف الاحتمالات بحيث يوضع للحاسب ما يجب عمله عند التعرض لتلك الاحتمالات .

٣ - في معظم الأحيان ، وخاصة عند حل مشاكل كبيرة تتطلب مجموعة كبيرة من الأوامر والتعليمات ، فإنه من المفضل جعل الحاسب يكتب نتائج وسيطة عند مواضع مختلفة من البرنامج بحيث يكون من السهل معرفة موضع بداية الخطأ في الحسابات ، إذا ظهر أن النتيجة النهائية غير صحيحة وهذا يوفر كثيراً من الوقت والجهد في البحث من أول البرنامج الى موضع ذلك الخطأ ، وسنلمس ذلك عندما نتمكن من كتابة برامج تشمل عشرات أو مئات الأوامر والتعليمات .

٤ - أن نضع في أذهاننا وألا ينبغي عنا ، أن الحاسب ماهو إلا أداة حسابية منفردة بكل دقة وبدون تفكير ، للأوامر والتعليمات التي يتلقاها ، وكلما كانت تلك الأوامر

لتبين للحاسب

Subprograms

(٥) نموي الحاسبات بعض البرامج الجارية

طريقة حل بعض الدوال الخاصة مثل إيجاد الجذر التربيعي لقيمة ما وحساب لوغاريتم عدد ما أو قيمته المطلقة أو ... الخ . وعلى سبيل المثال فإنه لحساب قيمة متغير مرفوع الى قوة ما فإن ذلك يتطلب حساب متسلسلة أسية Power Series متقاربة يلزم لحسابها عمليات ضرب تفوق كثيراً العمليات التي تلزم لها الطريقة (ب) ، كما أن قيمة تلك المتسلسلة ستكون تقريبية ( يمكن الرجوع في ذلك الى كتب التحليل المتدي ) .

والتعليمات واضحة وشاملة ومطابقة لقواعد اللغة التي يتعامل بها ، كلما أمكننا توفير الجهد والوقت والتوصل الى النتيجة الصحيحة .

#### ١-٤ المراحل المختطفة التي يمر بها برنامج على الحاسبات الالكترونية :

كلنا يتقن اللغة العربية قراءة وفهما وكتابة وتلم الماسماً تماماً بفهم قواعدها ، ولذا لانواجه بأية عقبات عند قراءتها أو فهمها أو كتابتها . والبعض منا لا يتقن الإنجليزية مثلاً ، وبالتالي سيواجه بعقبات وصعوبات عند قراءتها أو كتابتها لأنه لا يعرف قواعد تلك اللغة المعرفة الكافية والتي قد تسبب له الوقوع في أخطاء كثيرة وربما في التوقف كلية عن القراءة أو الكتابة . ولكن إذا أعطينا قاموساً لمساعدتنا في قراءة ومعرفة قواعد تلك اللغة فلن تكون هناك مشكلة أو صعوبة في قراءتها أو كتابتها ، وينطبق نفس الشيء على أية لغة أخرى لانجهدنا .

وكما سبق أن علمنا أن الحاسبات ماهي الا أدوات تنفيذ لأوامر وتعليمات بسرعة كبيرة طالما أن تلك الأوامر والتعليمات واضحة لها ومطابقة لقواعدها . فأذا اردنا أن نتعامل مع تلك الحاسبات فما علينا الا أن نعطياها أوامر وتعليمات وبيانات المشكلة التي نريد حلها باللغة التي تجهدها ، الا وهي لغة الحاسب نفسه Machine Language . ونظراً لأن الحاسبات تعمل بدوائر الكترونية Electronic Switching Circuits يتحكم فيها عنصرى حساب النظام الثنائي The Binary System أي الصفر والواحد ، ولذا فإن اللغة المباشرة والتي يمكن التعامل بها مع الحاسبات هي تلك التي تحتوي على هذين العنصرين فقط . ونظراً للصعوبة الكبيرة التي ستواجهنا في التعامل مع الحاسب بلغته ، فقد أمكن عمل مايشبه قواميس للغات مختلفة يمكننا الكتابة بها ويقوم الحاسب بمساعدة تلك القواميس الى ترجمتها الى لغته . ويطلق على تلك القواميس أو الترجمات اسم المصنفات أو المجمعات Compilers .

ولذا فلنكني يتمكن الحاسب من فهم لغة ما غير لغته لا بد أن يحولها أولاً الى لغته . ولكي ينفذ التعليمات المطاه له في صورة برنامج فلا بد أن تعطى تلك التعليمات للحاسب بشرط أن تكون متفقة مع قواعد اللغة التي ترجمها الى لغته ومن هنا تأتي أهمية دراسة القواعد والشروط التي لا بد من اتباعها لكتابة برنامج بلغة غير لغة الحاسب .

وفي بدء ظهور الحاسبات كانت هناك لغة واحدة معروفة قريبة من لغة الحاسب تسمى Assembly Language وكانت تترجم الى لغة الحاسب باستخدام برنامج وسيط يسمى Assembler . ونظراً لصعوبة تلك اللغة فقد استحدثت لغات أخرى بعضها يفوق الآخر في تطبيقات علمية معينة وعددها الآن يزيد عن خمسمائة لغة أهمها وأكثرها شهرة واستخداما هي اللغات :

## FORTRAN - BASIC - COBOL - ALGOL - PASCAL - PL1.

وكل لغة من تلك اللغات لها قواعدها وامكانيات تطبيقها في مجالات علمية وعملية معينة ، فعل سبيل المثال فأن أكثر مجالات استخدام لغة الفورتران FORTRAN هي في الحسابات العددية والمهندسية ، بينما تستعمل لغة الكوبول COBOL غالباً في الحسابات التجارية . وهذا لا يمنع استخدام اللغات الباقية في المجالين السابقين ولكن سيكون ذلك بامكانيات وكفاءة تقل عن اللغتين السابقتين . ويطلق على تلك اللغات اسم اللغات ذات المستوى العالي High level language .

ولذا فقبل البدء في تنفيذ برنامج بلغة ما على الحاسب لابد من التأكد أولاً أن الحاسب يحوي في ذاكرته ( وحدة التخزين ) المصنف الخاص بتلك اللغة والا فأن الحاسب لن يستطيع فهمها وبالتالي ترجمتها وتنفيذها . وفي عصرنا الحالي فأن الحاسبات الكبيرة يمكنها التعامل مع أكثر من لغة في وقت واحد نظراً لأن مصنفات تلك اللغات يكون قد سبق وضعها في وحدات تخزين تلك الحاسبات .

وعند حل مشكلة ما على الحاسب فإنه لابد من المرور بالمراحل التالية :

- ١ - تحضير خطوات حل المشكلة بعمل رسم تخطيطي لها يبين انسياب تلك الخطوات ومدى تسلسلها وترابطها مع بعضها البعض . ويطلق على هذا الرسم التخطيطي اسم « مخطط تدفق الخطوات الحسابية والمنطقية » FLOW CHART . وسيأتي الحديث عنها بالتفصيل في الفصل الثاني .

- ٢ - كتابة برنامج حل المشكلة ( بالاستعانة بمخطط التدفق اذا تطلب الأمر ذلك ) بلغة يكون قد سبق التأكد من أن الحاسب يتعامل معها ويستطيع ترجمتها ( وسنقصر حديثنا في هذا الكتاب على لغة الفورتران وقواعدها ) .

- ٣ - يغذى الحاسب بالبرنامج عن طريق وحدة من وحدات ادخال التعليمات والبيانات .

- ٤ - يقوم الحاسب باختيار مدى مطابقة التعليمات الواردة في البرنامج بقواعد اللغة التي كتب بها ، واذا كانت هناك اخطاء فأن الحاسب يكتشفها ويجب تصحيحها أولاً قبل البدء في ترجمة تلك التعليمات .

- ٥ - بعد تأكد الحاسب من أن البرنامج لا يحتوي على اخطاء خاصة بقواعد اللغة التي كتب بها يبدأ في ترجمة البرنامج الى لغته ( أي لغة الحاسب ) . وتسمى المرحلتين ٤، ٥ بعملية التصنيف Compilation والتي تشمل بداخلها عمليات :

الترجمة Translation والتجميع Assembly وتكوين التركيب Structuring .

- ٦ - بعد ذلك تبدأ مرحلة التنفيذ Execution ويقوم فيها الحاسب بتنفيذ الأوامر بالترتيب الذي كُتبت به في البرنامج بعد أن تكون قد ترجمت إلى لغة الحاسب .
- ٧ - في مرحلة التنفيذ ، قد يكتشف الحاسب نوعاً آخر من الأخطاء أثناء تنفيذ بعض العمليات الحسابية مثل عملية قسمة تحوي في مقامها صفراً أو كمية سالبة يراد حساب جذرها التربيعي لها أو لوغاريتم لكمية سالبة وهكذا . وبعض الحاسبات تعطي تحذيراً مكتوباً على وحدة استخراج النتائج المستخدمة عند اكتشافها لمثل هذا النوع من الأخطاء وقد يظهر ذلك التحذير أيضاً في صورة ضوئية (إضاءة لمبة) على وحدة التحكم ، وقد يتوقف تنفيذ البرنامج بعد ذلك لأصلاح الخطأ أو يستمر في تنفيذ باقي تعليمات البرنامج ولكن بنتائج وسيطة مشكوك فيها . ولذا فإنه يفضل قبل تنفيذ برنامج على الحاسب أن يتم حساب المشكلة يدوياً بمثال بسيط ويتم تنفيذ البرنامج مستخدمين نفس البيانات البسيطة فإن تطابقت النتيجة أمكن استخدام البرنامج بإطمئنان لحل مشاكل كبيرة .

## الفصل الثاني

### الثوابت والمتغيرات والعمليات الحسابية



## الفصل الثاني

### مقدمة :

قد يتساءل القارئ لأول وهلة بعد الانتهاء من قراءة الفصل الأول عن الفارق بين الآلة الحاسبة The Calculator والحاسب الآلي The Computer ؟ . وقد يقول قائل لماذا يتجشم عناء استعمال الحاسب الآلي ونحوض في تعقيدات لاطال منها طالما أن الآلة الحاسبة تستطيع القيام بنفس المهام وبالتالي فأنا نستطيع توفير كثير من الجهد والوقت والمال حيث أن الآلة الحاسبة لا تكلف الا مبلغا يسيراً من المال وبالتالي فهي في متناول الجميع . وفي الواقع فأن كل هذه الأسئلة وغيرها تعتبر أسئلة منطقية ومعقولة ولكن هذا التساؤل يتلاشى ويذوب عندما نستعرض النقاط التالية .

أولاً : أن جزءا كبيرا من استعمال الآلة الحاسبة يتم بطريقة ميكانيكية يدوية عن طريق الضغط على الأزرار المختلفة ، وبالتالي فأن على مستعمل الآلة الحاسبة أن يقوم بكل الخطوات العملية واحدة بعد الأخرى حتى ولو كانت هذه الخطوات مكررة ومملة وبفس الخطأ والأسلوب بينما يقوم الحاسب الآلي بنفس المهمة بطريقة آلية بحتة تتم بواسطة تنفيذ بعض الأوامر والتعليمات الصادرة اليه عن طريق برنامج مكتوب بلغة خاصة وطريقة خاصة .

ثانياً : إذا افترضنا أن سرعة اجراء العمليات الحسابية مثل الجمع والطرح ... الخ في الآلات الحاسبة تتساوى مع نظيرتها في الحاسبات الآلية الا أنه من المؤكد أن درجة الدقة تختلف اختلافا كبيرا . فبعض المشاكل الفيزيائية والكيميائية تتطلب درجة كبيرة من الدقة قد تصل الى عدد من الأرقام العشرية لاستطيع الآلات الحاسبة الوصول اليها بينما أصغر الحاسبات الآلية يمكن أن تتراوح نتيجة العملية الحسابية بها ما بين ١٠-٣٨ ، ١٠-٣٨١٠ . وعلى سبيل المثال فأن قيمة النسبة التقريرية في أي آلة حاسبة هي ٣١٤١٥٩٢٦٥٤ بينما يمكن حسابها بدقة أكبر بكثير في الحاسبات الآلية لتصل الى ٣١٤١٥٩٢٦٥٣٥٨٩٧٩٣٢٣٨٤٦٢٦٤٣٠٠٠ . وفي الحاسبات الآلية الكبيرة مثل CDC 6000/7000 يمكن أن تتراوح نتيجة العملية الحسابية ما بين ١٠-٣٢٢٠ ، ٣٢٢٠-٣٢٢٠٠ .

ثالثاً : أن لدى الحاسب الآلي المقدرة على التكرار Repetition . فهو لديه المقدرة على أن يقوم بالآلاف بل مئات الألوف من العمليات المكررة في ثوان معدودة بواسطة برنامج لا يتجاوز العشرة أسطر مثلا في حين أننا نحتاج الى عشرات أو مئات الساعات لانجاز نفس المهمة بواسطة الآلة الحاسبة ، والمقصود هنا بالعملية المكررة أي العملية التي تجري لأكثر من رقم ومثال ذلك إيجاد الجذر

التربيعي لعشرة أرقام موجبة مختلفة ، فالعملية هي إيجاد الجذر التربيعي وهي مكررة أما الأرقام والنتائج فهي متغيرة بتغير الأرقام .

مثال (٩) :

لنفترض أن لدينا قائمة تحتوي على عشرة آلاف رقم ، وطلب منا أن نوجد مربع كل رقم في هذا البيان .

### الحل

في الواقع هناك أكثر من حل لهذه المسألة الطويلة القصيرة ، ويمكن تلخيص الحلول المختلفة كالتالي :

١ - الحل اليدوي : وهذا الحل قد يستغرق اسبوعاً كاملاً خصوصاً إذا كانت الأرقام كبيرة وذات أجزاء عشرية كبيرة .

٢ - الحل باستخدام الآلة الحاسبة : وهي تلخص في إدخال كل رقم على حدة ثم إيجاد مربع الرقم وتسجيل الناتج بعد كل مرة وهذا يتطلب تكرار العملية عشرة آلاف مرة مما يستغرق جزءاً كبيراً في الوقت والجهد .

٣ - الحل باستخدام الحاسب الآلي : وهذا يتم عن طريق كتابة برنامج بسيط بلغة الفورتران أو أي لغة مماثلة ، يتكون من أربع أو خمس تعليمات بسيطة يرفق معه البيان بالكامل لنحصل بعد ثوان قليلة على بيان كامل يحتوي على جميع الأرقام ومربعاتها مطبوعة بشكل مرتب وأنيق .

ومما سبق يتضح لنا أن الاعتماد على الحاسب الآلي أصبح من البديهيات التي يقوم عليها عنصر السرعة والتكنولوجيا الحديثة التي تسمى الى التقدم المادي للإنسان لتوفر له مزيداً من الراحة والرفاهية والترفيه ، أما كيف ينفذ هذه الأوامر ، فيتم بواسطة تقديم هذه التعليمات والأوامر اليه في شكل خاص منظم يسمى بالبرنامج Program .

### تعريف

البرنامج : سلسلة من التعليمات المكتوبة بلغة خاصة وترتيب معين

ومما لاشك فيه أنه بمرور الوقت فسيذكر الطالب مدى القدرات الهائلة للحاسب الآلي مقارنة بالآلة الحاسبة فضلاً على أن بعض العمليات والمشاكل لا يمكن أن تحل بغير الحاسب الآلي ويستحيل حلها بواسطة الآلة الحاسبة العادية حيث يرجع ذلك الى قدرة الحاسب على تخزين كميات كبيرة من المعلومات واستخدامها فيما بعد بعكس الآلة الحاسبة ذات القدرات المحدودة البسيطة .






## « تقارين عامة »

١ - حاول أن تستفسر عن امكانيات الحاسب الموجودة في العمل ، كأن تسأل مثلاً عن عدد عمليات الجمع أو الضرب التي يستطيع أن يؤديها الحاسب خلال ثانية واحدة ؟ قارن الاجابة التي ستحصل عليها بالوقت الذي تحتاجه الآلة الحاسبة ؟

### ٢-١ التخطيط لكتابة برنامج :

عندما تواجهه الأنسان مشكلة من مشاكل الحياة العملية المعقدة فإنه بلا شك يحتاج الى كثير من التخطيط والتروي والتركيز وتقدير جميع الاحتمالات الممكنة ليضع لها الحلول المناسبة والتصورات الصحيحة ليعمل بموجبها ويسير على ضوئها وبالتالي ينجح في حل تلك المشكلة بالطريقة المناسبة .. ولحل مشكلة علمية بواسطة الحاسب الآلي يحتاج الأمر الى كتابة برنامج يؤدي الغرض المطلوب ، وكتابة البرنامج تحتاج الى دراسة وتخطيط وتركيز وتقدير مختلف الاحتمالات لوضع الحلول المناسبة . ولهذا فإن المبرمج يحتاج عادة الى عوامل مساعدة ليتمكن من كتابة البرنامج المطلوب .

ومن أهم هذه العوامل المساعدة استعمال مايسى FLOWCHART والذي يمكن ترجمة معناه الى عخطط التدفق للعمليات الحسابية والمنطقية والذي يمكن التفكير فيه كرسوم تخطيطي يوضح تسلسل التعليمات التي سيتكون منها البرنامج ليؤدي الغرض المطلوب . وكما أن للرسم المعماري رموز خاصة ترمز الى اجزاء معينة من المبنى فإن عخطط التدفق ايضاً بعض الرموز المتفق عليها والتي يدل كل منها على أمر معين . انظر الشكل (١-٢) .

الرمز	الدلالة	المعنى
	نقطة بداية أو نهاية	ترمز الى بداية أو نهاية البرنامج
	ادخال بيانات	ترمز الى ادخال بيانات رقمية أو غيرها من البيانات .
	إخراج بيانات	ترمز الى إخراج بيانات رقمية وغيرها .

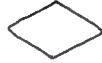
ترمز الى اجزاء عملية  
حسابية كالجمع والطرح  
... الخ .

عملية حسابية  
Process / Arithmetic  
Operation



ترمز الى عملية منطقية يُتخذ  
فيها القرار اللازم ليتمحدد  
على ضوءه اتجاه انسياب  
البرنامج .

عملية منطقية  
Decision / Logical  
Operation



الشكل (٢-١) رموز مخطط التدفق الانسيابي

اما اتجاه العمليات فسوف يحدد بواسطة استعمال الأسهم والتي تساعد على توضيح المخطط  
وتيسيطه للقارئ أو المبرمج .

مثال (٢) :

ارسم مخططاً تدفقياً يوضح سير العمليات الحسابية لبرنامج يقوم بمعالجة بيان يحتوي على رقم  
واحد فقط ليوجد مربع ذلك الرقم ومن ثم يكتب الناتج

الحل

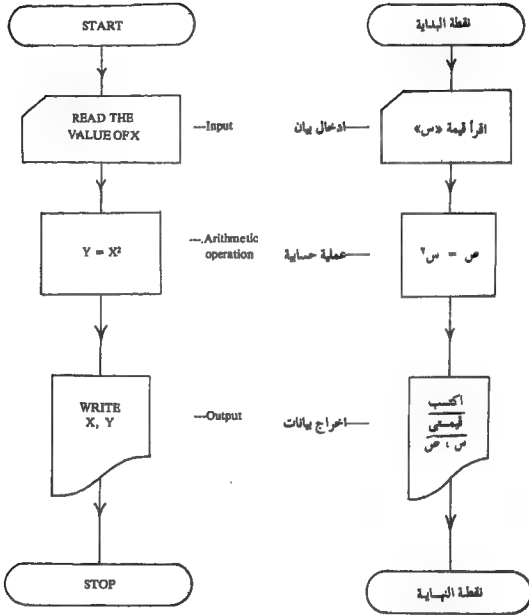
لو فكرنا قليلاً في الخطوات العملية لحل مشكلة بسيطة كهذه فأنا بلاشك سنصل الى اجماع عام  
بأن هذه الخطوات يجب أن تكون على الشكل التالي :

- ١ - نبدأ بقراءة الرقم المعطى ( ادخال بيان ) ولنرمز له بالحرف «س» .
- ٢ - نوجد مربع «س» ولنرمز له بالحرف «ص» ( عملية حسابية ) .
- ٣ - نكتب قيمتي «س» و «ص» ( اخراج بيانات ) .

هذه الخطوات المكتوبة البسيطة تساعدنا كثيراً في رسم مخطط التدفق للبرنامج والموضح في  
الشكل (٢-٢) .

من الملاحظ أن المخطط السابق قد وضع باللغتين العربية والانجليزية لتوضيح وتيسير الأمور في  
البداية ، ولكننا من الآن فصاعداً سنضع جميع مخططات التدفق باللغة الانجليزية فقط لأن كتابة  
البرامج ستكون بلغة الفورتران التي تكتب باللغة الانجليزية .

ولنحاول الآن أن نرسم مخطط التدفق لحل مشكلة أكثر صعوبة وتعقيداً من المشكلة السابقة .



الشكل (٢-٢) مخطط التدفق لعملية تربيع رقم

مثال (٣) :

لنفترض أن البيان المعطى في المثال السابق أصبح يتكون من مائة رقم بدلاً من رقم واحد وطلب منا أن نرسم مخططاً تدقيقياً يوضح سير العمليات الحسابية والمنطقية لحل مشكلة كهذه باستعمال الحاسب .

### الحل

ان من أكثر الطرق العملية والمنطقية للتفكير في حل مشكلة كهذه هي طريقة الآلة الحاسبة التي يمكن تلخيصها فيما يلي :

- ١ - ادخل الرقم الأول من البيان في الآلة الحاسبة .
- ٢ - اضغط زر التربيع أو اضرب الرقم في نفسه .
- ٣ - سجل مربع الرقم الناتج .
- ٤ - ادخل الرقم الثاني من البيان في الآلة الحاسبة .
- ٥ - كرر الخطوات الثانية والثالثة .
- ٦ - كرر الخطوات الثانية والثالثة لكل رقم من الأرقام المتبقية بعد ادخال كل رقم على حده حتى يتم استكمال البيان .

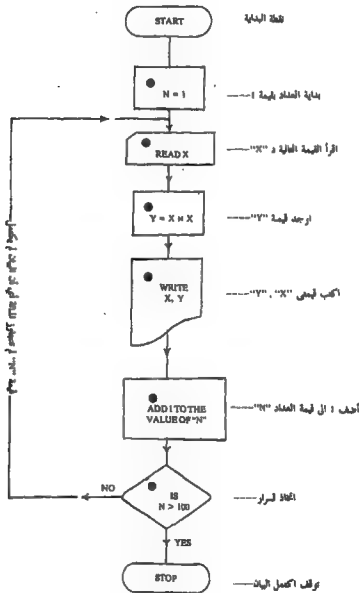
بما سبق يتضح لنا أن قدرتنا الطبيعية على احصاء عدد الأرقام التي تم معالجتها من البيان في وقت ما سوف يمكننا من التوقف بعد الانتهاء من البيان تماماً ، أي ليس من المعقول أن نتوقف بعد خمس و نسمون رقماً فقط . اذا فلانسان بطبيعته عدداً أو مؤشراً يساعده على التوقف عند الخطوة المناسبة وهذه الخاصية لا يمتلكها الحاسب الآلي لكنه من الممكن لنا أن نكسبه هذه القدرة على الاحصاء اذا ما استعملنا مؤشراً أو عدداً يمكنه من التوقف في الوقت المناسب ، وهذا ما سنضعه في الاعتبار عند رسم مخطط التدفق لهذا المثال .

أما الآن فلنكتب الخطوات العملية التي نريد تنفيذها في الواقع لحل المشكلة والتي يمكن ترتيبها كالآتي :

- ١ - نبدأ باعطاء العداد ولترمز له بالـ «N» القيمة المبدئية 1 .
- ٢ - نقرأ أول رقم في البيان ولترمز له بالـ «X» .
- ٣ - توجد مربع قيمة «X» ولترمز له بالـ «Y» .
- ٤ - نكتب قيمتي «X» و «Y» .
- ٥ - أضف مقدار واحد الى قيمة «N» السابقة لتصبح القيمة الحالية مساوية القيمة الأولى مضافاً اليها 1 .

- ٦ - تجري العملية المنطقية التالية : اذا تجاوزت قيمة «N» المائة فأنا نأمر الحاسب بالتوقف لأنه استكمل المطلوب منه . أما اذا لم تتجاوز قيمة «N» المائة فأنا عندئذ نقوم بقراءة القيمة التالية في البيان والتي تمثل القيمة الجديدة للمتغير «X» .
- ٧ - تكرر الخطوات من ٣ الى ٦ .

بعد هذا العرض الموجز للخطوات المطلوبة نقوم برسم مخطط التدفق للعمليات الحسابية والمنطقية كما هو موضح في الشكل (٣-٢) .



الشكل (٣-٢) مخطط التدفق لبرنامج لإيجاد مربعات مائة رقم

## « تمارين »

اكتب وصفاً مبسطاً بالإضافة الى مخطط التدفق لبعض العمليات التالية :

- اصلاح بنشر سيارة .
- شراء زوج من الأحذية .
- حساب معدل التراكبي في الجامعة .
- الحصول على رخصة قيادة خصوصية .
- حساب دخلك السنوي .
- الذهاب في رحلة الى البحر .

## ٢-٢ بعض العمليات الأساسية للحاسب الآلي :

بعد هذه المقدمة البسيطة عن مخطط التدفق للعمليات المنطقية والحسابية نصبح الآن في مرحلة نستطيع فيها أن نبدأ بكتابة برنامج بسيط بلغة الفورتران - ولكن قبل أن نبدأ ذلك نريد أن نتساءل عن العمليات والمهام الأساسية التي يستطيع أن يؤديها الحاسب الآلي عن طريق تعليمات وأوامر تشكل في مجموعها مايسمى بالبرنامج ؟ وفي الواقع فإن هذه العمليات يمكن تقسيمها الى ثلاث أنواع رئيسية هي :

**أولاً :** عمليات ادخال واخراج البيانات : وهذه احدى جوانب تفوق الحاسب الآلي فهو يقوم بمعالجة البيانات الواردة اليه مهما كانت ضخمة وطويلة . وتنتهي عادة هذه المعالجة ببيانات صادرة تمثل النتائج المطلوبة من البيانات الواردة بعد إجراء سلسلة من العمليات الحسابية والمنطقية عليها والتي تختلف من برنامج لآخر حسب الدور المطلوب منها . ففي المثال الأخير نلاحظ أن البيان الوارد يتكون من مائة رقم أما البيان الصادر فيتكون من مائتي رقم تشمل الأرقام الواردة ومربعاتها . وهاتين العمليتين تتمان عن طريق استعمال التعليمات .

« اقرأ » ( بيان وارد ) READ

« اكتب » « وأطبع » ( بيان صادر ) WRITE AND PRINT

ثانياً : العمليات الحسابية والمنطقية : وتشمل عمليات الحساب المعروفة من جمع وطرح وضرب وقسمة وأسس وغير ذلك من العمليات الرياضية كاللوغاريتمات والجذر التربيعي والدوال المثلثية ... الخ . أما العمليات المنطقية فهي تشمل عمليات المقارنة بين مقدارين أو أكثر ينتج على أثرها قرار معين يعتمد على نتيجة المقارنة فمثلاً قد تتم المقارنة بين مقدارين «س» و «ص» ومن ثم فإن قراراً

معيناً سوف يتخذ في حالة ما اذا كانت قيمة «س» أكبر من قيمة «ص» وقد يتخذ قراراً آخر مختلف تماماً في حالة ما اذا كانت قيمة «س» أصغر من قيمة «ص» وهكذا .

ثالثاً - عمليات التحكم : وهي تشمل العمليات التي تغير من المسار الطبيعي للبرنامج كما تشمل تعليمات التنفيذ المشروطة وتعليمه إيقاف البرنامج وغيرها . وستكلم فيما بعد عن هذه التعليمات بالتفصيل . فيما يلي سنحاول أن نستقرئ بعض العمليات المذكورة أعلاه من خلال البرنامج التالي :

## ٣-٢ برنامج حساب الاستحقاق السنوي :

مسألة :

اكتب برنامجاً يحسب الاستحقاق السنوي وكذلك صافي الاستحقاق السنوي لشخص يبلغ راتبه الشهري ٩٦٥٠ ريالاً سعودياً يدفع منها ٩٪ رسوم تقاعد كما يحصل على بدل مواصلات نقدي مقداره ٦٠٠ ريال سعودي .

### تقريين

أرسم مخططاً تدقيقياً يوضح مسار العمليات الحسابية والمنطقية للمسألة أعلاه

### الحل

الشكل (٤-٢) يمثل البرنامج المطلوب .

20

```
REAL SALARY, TRANSP, TAX, GROSS, NET
READ (5,*) SALARY, TRANSP, TAX
GROSS = 12.0 * SALARY + 12.0 * TRANSP
NET = GROSS - 12.0 * TAX * SALARY
WRITE (7,20) GROSS, NET
FORMAT (F 8.2, 5X, F7.2)
STOP
END
```

الشكل (٤-٢) البرنامج المطلوب لحل المسألة أعلاه

من الشكل السابق يبدو الأمر لأول وهلة وكأنه موضوع انشاء باللغة الانجليزية رغم أن المحتوى قد لا يعني الكثير بالنسبة لنا في هذه المرحلة ، ولكننا نستطيع أن نتلمس طريقنا خلال السطور وأن نفهم بعض ما يدور في هذا البرنامج البسيط رغم أننا لا نملك القدرة حالياً على كتابة أي برنامج

فورتران . على أي حال سوف نحاول من خلال الصفحات القليلة القادمة أن نشرح كل جملة من جمل هذا البرنامج على حدة بصورة مبسطة تعطي انطباعاً عاماً عن شكل البرنامج المكتوب بلغة الفورتران .

فالسطر الأول يُعلم الحاسب الآلي بجميع أسماء المتغيرات التي سوف تستعمل في البرنامج فمثلاً استعملت كلمة SALARY كاسم للمتغير الذي يمثل الراتب الشهري للشخص بينما استعملنا كلمة TRANSP ( اختصار كلمة TRANSPORTATION ) لترمز للمتغير الذي يمثل بدل المواصلات الشهري وكذلك الحال بالنسبة للمتغير GROSS الذي يعني إجمالي المستحق و TAX الذي يمثل ضريبة التقاعد ، أما المتغير NET فيمثل صافي الاستحقاق السنوي . وللملاحظة فإن هذه القائمة تسبقها كلمة REAL وهي توضح للحاسب أن مقادير هذه المتغيرات يجب أن تكون أرقاماً ذات فواصل عشرية ، وسوف نتكلم عن هذا الموضوع بإسهاب في حينه .

أما السطر الثاني فهو عبارة تعليمة لأدخال بيان يشمل ثلاث مقادير ثلاث متغيرات هي SALARY ، TRANSP ، TAX ويم هذا عن طريق كلمة READ كما أسلفنا سابقاً .

أما السطرين الثالث والرابع فهي عمليات حسابية بسيطة ينتج على أثرها حساب قيمتي المتغيرين GROSS و NET بمعرفة مقادير المتغيرات الأخرى والتي عُرفت عن طريق ادخال البيان في السطر الثاني . يلي ذلك تعليمة لأخراج بيان يشتمل على قيم المتغيرات المطلوب حسابها وهي كما نعرف من السؤال تمثل حلاً للسؤال المعطى ، والتعليمة تبدأ بكلمة WRITE والتي تعني كما أسلفنا « إكتب » ثم تكتب قائمة بأسماء المتغيرات المطلوب كتابة قيمها حسب ترتيبها في تعليمة الكتابة وحسب الطريقة الموضحة في الجملة التالية والتي تحمل الرقم 20 في بدايتها مع ملاحظة أن هذا الرقم موجود في تعليمة WRITE السابقة لجملة FORMAT والتي تعني في العربية « هيئة » أو « صفة » أو « طريقة » .

أما السطر التالي الذي يشتمل على كلمة STOP والتي تأمر الحاسب بأيقاف التنفيذ فوراً وعدم القيام بتنفيذ أي تعليمة من الآن فصاعداً ، وفي النهاية فإن كلمة END تعني النهاية الحسية للبرنامج حيث لا يمكن كتابة أي تعليمة بعد ذلك .

وفي الواقع فإنه يتضح لنا بعد مراجعة العمليات الأساسية التي يمكن أن يقوم بها الحاسب الآلي أن هذا البرنامج يعتبر نموذجاً بسيطاً يشمل أمثلة مختلفة من هذه العمليات الأساسية .

أما الآن وقد أشبعنا فضول الطالب المهتم عن شكل البرنامج ومحتوياته والتي قد يتخيلها الطالب لأول وهلة أنها لا بد وأن تكون جمل وكلمات مسحورة عجيبة توجه الى الحاسب الآلي فيتصرف أذاؤها بتعقل وحكمة تفوق تعقل وحكمة أي كائن بشري ولكن الواقع أن الحاسب الآلي أغشى من



أي مخلوق في الوجود فهو عبارة عن آله صماء لا تملك حولاً ولا قوة ولا تخيد تفكيراً أو تدبيراً . ولذلك كان لابد من اعطاء هذه التعليمات بدقة بالغة فأبى اختلال في هذا التعليمات حتى ولو كان خطأً املأها فإنه قد يؤدي الى توقف البرنامج كلية دون تنفيذ أي جزء منه وفي أحسن الظروف قد يؤدي هذا الخطأ الى الحصول على نتائج خاطئة وربما مضحكة وغير معقولة .

لذا فأنتنا ننصح الطالب الراغب في فهم لغة الفورتران أو أي لغة حاسب أخرى أن يكون دقيقاً حريصاً واعياً مدركاً لحقائق عمل البرنامج وقواعد اللغة التي كتب بها وكيفية تنفيذ الحاسب لخطواته حتى يستطيع تلافي كثير من الأخطاء الغير ضرورية والتي تضيع كثيراً من الجهد والوقت والمال .

#### ٢-٤ طريقة كتابة برنامج ما :

أما كيفية تقديم البرنامج الى الحاسب الآلي ليقوم بمهمة تنفيذه واطهار النتائج المتوقعة منه فيم عن طريق تقديم البرنامج بإستعمال الوحدة الطرفية Terminal وهي أكثر الطرق شيوعاً الآن لكونها عملية ومريحة واقتصادية في ذات الوقت . وقد تختلف طرق إستعمال الجهاز من شركة الى أخرى ولكنها كلها تدور حول نفس الفكرة الأساسية التي يمكن تلخيصها فيما يلي :

١ - تشغيل الجهاز	SWITCH ON
٢ - الدخول في النظام	LOG ON
٣ - كتابة البرنامج	WRITING PROGRAM
٤ - تقديم البرنامج	SUBMISSION
٥ - تنفيذ البرنامج	EXECUTION

وجميع هذه الخطوات ماعدا الخطوة الثالثة يمكن اعتبارها أمور فنية تختلف من نظام الى آخر ولهذا فلن نخوض في تفاصيلها بل نتركها لأستاذ المادة ومساعدته الفني والنظام المستعمل في الجامعة أو المعهد الذي يتبعه الطالب .

أما كتابة البرنامج فتكاد تكون متائلة في جميع الأنظمة تقريباً ولهذا فسنناولها هنا ببعض التفصيل لأهميتها . ففي لغة الفورتران يتكون السطر الواحد من ٨٠ خانة ( أو ١٢٠ خانة في بعض الأجهزة ) إلا أن الخانات الفعلية المستعملة هي الخانات من ١ الى ٧٢ وهذه الخانات موزعة الى مناطق عمل رئيسية على الوجه التالي :

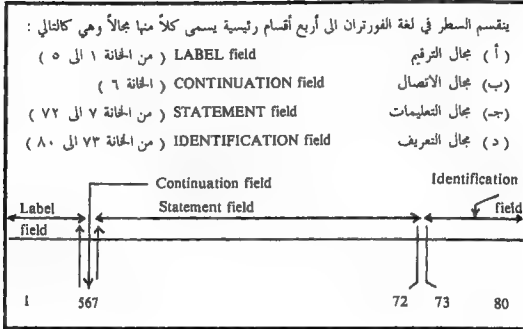
- الخانات من ١ الى ٥ تستعمل لتحديد رقم الجملة ( إن وجدت الحاجة الى ذلك ) ويسمى هذا المجال « مجال الترميم » LABEL field ، ويمكن وضع أي رقم موجب صحيح في هذا المجال . ولا يمكن جملتين مختلفتين أن تحملان نفس الرقم .

- الخانة رقم ٦ تترك عادة خالية BLANK إلا في حالة واحدة فقط هي حالة كون المعلومات المكتوبة في الخانات ٧ الى ٧٢ تعتبر مواصلة للجملة التي تسبقها في السطر السابق . أي أن وجود أي رمز CHARACTER ماعدا الفراغ أو الصفر في الخانة ٦ يعني أن السطر الحالي ماهو الا امتداد للسطر السابق لأن السطر السابق لا يتسع للجملة بأكملها . ويسمى هذا السطر CONTINUATION LINE « السطر المتواصل » . ولا يجوز شغل « مجال التعريف » في حالة شغل الخانة ٦ لأن رقم الجملة هو نفس رقم الجملة السابقة بإعتبارها اتصالاً لها . وسوف نرى في أمثلة قادمة كيف أن بعض الجمل تشغل أكثر من ٦٤ خانة (٦٦ - ٧٢) .
  - الخانات من ٧ الى ٧٢ هي الخانات التي توضع فيها التعليمة أو الجملة طبقاً لقواعد لغة الفورتران ويسمى هذا المجال بمجال التعليمات « STATEMENT field » .
  - يجوز استخدام الخانات من ٧٣ الى ٨٠ لأغراض أخرى غير ذات علاقة بمضمون الجملة المكتوبة في الخانات من ٧ الى ٧٢ . وأهم هذه الأغراض هي المعلومات التعريفية الإضافية للبرنامج .
  - في حالة وضع الحرف C في الخانة الأولى ينتج لدينا مايسمى « بستر الملاحظات » « COMMENT Line » وهذا السطر يتجاهله الحاسب الآلي تماماً من الناحية التنفيذية ، لكنه يقوم بطابعته فقط عندما يكتب البرنامج . ولهذا فيمكن للمبرمج أن يستعمل هذه الميزة الهامة بشرح بعض خطوات البرنامج شرحاً إملالياً لغوياً دون الخضوع لقواعد لغة الفورتران المحدودة ، وذلك عن طريق كتابة حرف C في الخانة الأولى كما أسلفنا ثم كتابة أي مادة يرغبها المبرمج في الخانات من ٢ الى ٨٠ .
- فمثلاً نستطيع أن نبدأ البرنامج المعطى في الشكل (٢-٤) بالسطور التالية :

#### الخانة الأولى

```
C *** THIS PROGRAM ISMADE BY A BEGINNER *$*+*
C
C
C
C . . . THIS PROGRAM COMPUTES THE ANNUAL SALARY AND THE
C . . . . . NET ANNUAL SALARY OF A GOVERNMENT
C . . . . . EMPLOYE.
```

ويمكن تلخيص ماسبق في المستطيل التالي :



## ٢-٥ تسمية المتغيرات :

كما سبق يتضح لنا أن المهمة الرئيسية للبرنامج هي معالجة البيانات تلقائياً وفق التعليمات المعطاه له وهذه الخاصية من أهم ما يميز الحاسب الآلي عن الآلة الحاسبة ، كما علمنا من الفصل السابق أن البيانات الواردة تخزن أولاً في الذاكرة ثم يقوم الحاسب بمعالجة هذه البيانات طبقاً للتعليمات الصادرة اليه عن طريق البرنامج لينتج لنا في النهاية البيانات الجديدة المطلوبة . ففي المثال السابق كان البيان الوارد يشتمل على قيم ثلاث متغيرات هي TAX, TRANSP, SALARY وهذه القيم تحفظ في الذاكرة تلقائياً حتى يتسنى لنا استعمالها فيما بعد في عمليات حسابية أخرى أو أن يُطلب كتابة بيان يشتمل على قيم هذه المتغيرات كما فعلنا في السطر الخامس من البرنامج .

إذاً الذاكرة جزء مهم من الحاسب الآلي ، لذا وجب علينا أن نضع تصوراً منطقياً وواقعياً عن طريقة تخزين البيانات في الذاكرة ومعالجتها وتخزين النتائج فيها لاستعمالها فيما بعد ، وأبسط هذه التصورات هي تصور الذاكرة كوحدة تشتمل على أعداد هائلة من الخلايا CELL المتماثلة تكون في بادئ الأمر بدون أسماء . وعند اطلاق اسم على متغير فإن الحاسب يقوم تلقائياً بتعيين خلية في الذاكرة تحمل ذلك الاسم باستمرار حتى انتهاء تنفيذ البرنامج . فمثلاً في المثال السابق كانت هناك خلية باسم SALARY و خلية باسم TRANSP وخلايا بأسماء TAX, NET, GROSS وهذه

الحلایا تحمل هذه الأسماء بصفة دائمة حتى انتهاء البرنامج وفي الواقع فإنه لا بهما موقع الخلية في الذاكرة بقدر ما بهما معرفة أن هناك خلية ما تحمل اسماً معيناً ثابتاً ، ولكن متى تكون الكلمة صالحة لأن تكون اسماً لخلية من حلایا الذاكرة ؟ فكما أن اختيار اسم الطفل المسلم يخضع لشروط معينة فكذلك الحال بالنسبة لتسمية الخلية ، أما هذه الشروط فهي كالآتي :

- ١ - أن يبدأ الاسم بحرف من حروف اللغة الإنجليزية (A - Z) .
- ٢ - أن يتكون الاسم من حروف (A - Z) أو أرقام (0 - 9) فقط .
- ٣ - لا يجوز أن يتكون الاسم من أكثر من ستة رموز . ومقصد بالرموز مجموعة الحروف أو الخليط من الحروف والأرقام بشرط أن يتحقق الشرط ١ .

مثال (٤) :

الأسماء التالية تعتبر صحيحة وشرعية في لغة الفورتران :

A, X, TF, N3, ALI, Z2B4, JAMAL, USA6, EGYPT

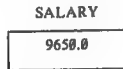
مثال (٥) :

لا يجوز استعمال الأسماء التالية في لغة الفورتران ، لماذا ؟

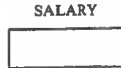
MAHMOOD, 4X, C4 BF, MAJIDS, X+T, U.S.A.

٢-٢ قيمة المتغير :

لنفرض أن لدينا خلية تحمل الاسم SALARY ولنتصور أن الخلية لها شكل مستطيل ( أنظر الشكل (٢-٥) أ ) .



الشكل (٢-٥) ب



الشكل (٢-٥) أ

ولنفترض أن الحاسب أعطى قيمة SALARY عن طريق قراءة بيان يشتمل على قيمة SALARY ولنفرض أن هذه القيمة هي 9650.0 . فالذي يحدث هو أن هذه الخلية التي تحمل الاسم SALARY تحتوي في الواقع بداخلها على القيمة العددية 9650.0 ( أنظر الشكل (٢-٥) ب ) . إذا يمكننا أن نشبه الخلية بكأس فارغ يحمل اسماً معيناً ثم نضع فيه ماءً مثلاً ليكون لدينا كأساً له اسم معين

ويحتوي على سائل هو الماء . ولكن من الواضح أنه يمكننا إستبدال محتوى الكأس من الماء بسائل آخر وليكن عصير برتقال مثلاً وفي هذه المرحلة بالذات نرى أن اسم الكأس لم يتغير بينما تغير محتوى الكأس تماماً ، وكذلك الحال بالنسبة للخلية التي تحمل اسماً معيناً خلال تنفيذ البرنامج - فالأسم لايمكن تغييره بينما يمكن تغير محتوى الخلية بأحدى التعليمات المناسبة .

مثال (٥) :

الشكل (٦-٢) يمثل التغير الحاصل في محتوى خلية تحمل الاسم X .

X	X	X
- 4.00	10.00	

( شكل (٦-٢) )

**قاعدة هامة :** خلال تنفيذ برنامج معين لايمكن تغير اسم خلية ما بينما يمكن تغير محتوى الخلية أي أنه لايمكن تغير اسم المتغير بينما يمكن تغير قيمته حسب تعليمات البرنامج .

ومن المهم أن ننوه هنا بأن الحاسب يقوم بمحجز خلية لكل متغير يخالف في اسمه للمتغيرات التي سبق ظهورها في البرنامج . فعلى سبيل المثال فإن المتغيرات .

B12A, AB21, AB12,A21B, A12B, BA21, BA12, B21A

وان تشابهت في مجموعة الرموز المكونة لأسمائها إلا أن كلا منها تختلف في اسمها عن الآخر مما يترتب عليه قيام الحاسب بمحجز خلية لكل منها .

## ٢-٧ أنواع قيم المتغيرات : Data types

هناك نوعان مختلفان من القيم العددية التي يمكن أن تخزنها خلية تحمل اسم متغير ما ، وهما كالتالي :

### أولاً : الأرقام الحقيقية ( العشرية ) : REAL numbers

وهي الأرقام التي تحتوي على نقطة عشرية واحدة وقد تكون موجبة أو سالبة . ويطلق عليها أيضاً اسم « الأرقام ذات العلامة العشرية المتحركة » Floating-point numbers .

### ثانياً : الأرقام الصحيحة : INTEGER numbers

وهي لا تحتوي أي اجزاء عشرية كما لا تحتوي أي نقاط عشرية ، أما أن تكون سالبة أو موجبة .

مثال (٦) :

تعتبر الأرقام التالية أرقاماً حقيقية بالنسبة للغة الفورتران .

1. 1.00 - 61.20 + 517.352 - 1146.9 0001.

بينما لا يجوز استعمال الأرقام التالية كأرقام حقيقية في لغة الفورتران .

1/2 - 1 + 1/5 639 - 78 1,146.9

مثال (٧) :

تعتبر الأرقام التالية أرقاماً صحيحة في لغة الفورتران :

1 156 - 034 + 85 - 1469

بينما لا يجوز معاملة الأرقام التالية على أساس أنها أرقاماً صحيحة في لغة الفورتران :

1. 47.0 - 4/2 + 85. - 1,469

لاحظ أن 1 و 1. يختلفان تماماً في نوعيتهما فالأول رقم صحيح والثاني رقم حقيقي مع أنهما لا يختلفان رياضياً .

## ٢-٨ الجملة المبينة ( التوضيحية ) : Declaration Statement

أما السؤال المتوقع الآن فهو كيف يمكن تحديد نوعية قيمة المتغير ؟ واختصاراً سوف نتكلم عن نوع المتغير نفسه فنقول متغير حقيقي ( عشري ) REAL Variable أو متغير صحيح INTEGER Variable والجواب على هذا السؤال يكمن في استعمال ما يسمى بالجملة المبينة ( التوضيحية ) Declaration Statement التي تبين أو تحدد نوع المتغير بالنسبة للحاسب الآلي ويمكن تلخيص عمل هذه الجملة في المستطيل التالي :

الجملة المبينة	Declaration Statement
الشكل العام	TYPE list
أما كلمة TYPE فهي إما أن تكون REAL أو INTEGER بينما List تعني قائمة بأسماء المتغيرات مفصولة بفواصل «،» وهذه القائمة تتكون من متغير واحد أو أكثر .	
المعنى : هذه التعليمة توجه الحاسب الآلي الى اطلاق الأسماء الموجودة في List على خلايا تستعمل لتخزين بيانات عددية من نوع TYPE المذكور في أول الجملة .	

مثال (٨) :

REAL A, B3, X8Z61, NUMBER, FOUR4  
 INTEGER ALPHA, COUNT, N, M1, FX24Y  
 INTEGER EGYPT, SP, TABLE, L3  
 REAL SAUDIA, R4, KARL, MTXF, LONDON  
 INTEGER SIX

فالسطر الأول يعتبر بمثابة تعليمة للمحاسب الآلي لاطلاق الأسماء الخمسة الموجودة في القائمة على خلايا ذات محتوى عشري ، أي أن A يعتبر متغيراً ذو قيمة عشرية وكذلك الحال بالنسبة لـ FOUR NUMBER, X8Z61, B3 أما السطر الأخير فيؤكد أن الحاسب سيعامل المتغير SIX على أساس أنه رقم صحيح . وماقبل عن السطر الأول والأخير يقال تماماً عن بقية الأسطر . ولكن علينا أن نضع في الاعتبار أن المتغير ذو القيمة العشرية يجب أن ينظر إليه على أساس أنه متغير حقيقي « عشري » طوال البرنامج وكذلك الحال بالنسبة للمتغيرات الصحيحة .

« تمارين »

١ - أي من التالي يمكن إعتباره اسماً شرعياً لحلقة ذاكرة ؟ علل في حالة النفي !

23K5  
 ILOVE7  
 ABDULLAH  
 I LOVE  
 K. A. U.  
 FORTRAN

٢ - أي من الجمل التالية يمكن إعتباره جملة « مبنية » في لغة الفورتران ؟ علل في حالة النفي !

INTEGER A  
 INTEGER A, BOB, X, IN  
 REAL K, JO, XY4  
 REAL ALPHA, 14.6  
 INTEGER VERYLONG, SHORT  
 REAL, A

## INTERGER BOX

## REAL INTGER

٣ - اكتب جملة (جمل) فورتران تبين أننا نريد أن نستعمل خليتي ذاكره باسم ALI و NADER لتخزين ارقام صحيحة (INTEGERS) !

٤ - فيما يلي ضع اشارة (✓) بجانب الرقم الحقيقي وضع دائرة حول الرقم الصحيح ؟

41.7

86

784.

- 49

+ 896.673

■

## ملاحظات هامة :

١ - المقادير الثابتة تنقسم الى فئتين هما فئة الأرقام العشرية REALS وفئة الأرقام الصحيحة

## . INTEGERS

٢ - المقدار المتغير يرمز له باسم يخضع لشروط التسمية في لغة الفورتران كما أوضحنا سابقاً ، وهذا لا يعني بالضرورة تغيير قيمة المتغير خلال البرنامج ، فمن المحتمل أن يكون للمتغير قيمة ثابتة لا تتغير طوال البرنامج .

٣ - لايحوز وضع اشارتين رياضيتين متعاقبتين بدون فصلها بواسطة قوس . فمثلاً :

5.0\*\*2

أو 3\*-2

لايحوز كتابة

5.0\*\*(-2)

أو 3\*(-2)

بينما يجب كتابتها على نحو

٤ - في حالة استعمال أرقام عشرية وأرقام صحيحة في نفس المقدار الجبري فإن الناتج يكون رقماً عشرياً .

4.\*\*2+3

فمثلاً ناتج المقدار الجبري

19.0

يساوي

## MIXED MODE OPERATION

وهذا ما يسمى بـ

٥ - لايحوز رفع رقم سالب الى أس عشري مثل 1.5\*\*(3.0) أو 2.0\*\*(-2) بينما يحوز رفع رقم

موجب الى أس عشري مثل 2.3\*\*(0.5) أو (-2.0)\*\*(1.67) والسبب في ذلك أن طريقة

ايجاد الحاسب لمقدار مثل (2.5)\*\*(4.15) تتم طبقاً للخطوات التالية :

- ايجاد لوغارهم الرقم العشري 4.15 أي  $\ln 4.15$



- إيجاد حاصل ضرب الأس 2.5 في المقدار  $\ln 4.15$  أو أوجد  $\ln 2.5$   
 - إيجاد حاصل قيمة رفع العدد الطبيعي  $e$  الى الناتج السابق لنحصل على النتيجة المطلوبة .  
 أي أن

$$(4.15)^{2.5} = e^{(2.5 \ln 4.15)}$$

وحيث أنه لا يمكن حساب لوغاريتم أي رقم سالب لذلك كان من غير الممكن تكليف الحاسب بإيجاد قيمة رقم سالب مرفوع الى أس عشري .  
 ٦ - حاول قدر المستطاع استعمال الأس الصحيحة بدلاً من الأس العشرية ، وذلك لدقة استعمال الأس الصحيحة لأنها تعني ضرب الرقم في نفسه عدد مرات يساوي قيمة الأس بينما الأس العشري يؤدي بالحاسب الى استعمال الطريقة المذكورة في ٥ . إذاً هناك فارق جوهري في طريقة إيجاد قيمة كل من :

$$(3.0)^{2.0} \quad \text{و} \quad (3.0)^{2.0}$$

بالرغم من أن الناتج يساوي 9.0 في كلا الحالتين ، ولكن الوضع سيختلف في حالة إيجاد أسس لبعض الأرقام العشرية المعقدة فمثلاً ، في حالة إيجاد قيم كل من :

$$(35.6324032)^{2.0} \quad \text{و} \quad (35.6324032)^{2.0}$$

سيكون هناك فارق في الناتج بالتأكيد لأن الأس الصحيح يعني ضرب المقدار في نفسه بينما الأس العشري يعني إيجاد لوغاريتم الى تقريب معين ومن ثم رفع العدد  $e$  الى ضعف قيمة اللوغاريتم مما يؤدي الى اعطاء قيمة تقريبية مغلفة عن الحالة الأولى .

## ٢-٩ موقع الجملة المبينة :

أما موقع الجملة المبينة فيجب أن يكون في بداية البرنامج أي في السطر الأول اذا كانت لدينا جملة واحدة فقط وإذا كانت هناك أكثر من جملة فيجب أن نكتب في السطور التالية مباشرة ، وليس هناك مانع من كتابة أي عدد من الجمل المبينة حتى وإن كان بالامكان الاستغناء عن بعض هذه الجمل فمثلاً يمكننا استبدال هاتين الجملتين الصحيحتين .

REAL X

REAL Y

REAL X,Y

بجملة واحدة صحيحة هي :

والواقع أن موقع الجملة المبينة في بداية البرنامج هو موقع طبيعي جداً حيث أن الجملة تحدد نوع المتغير الذي سوف يستعمل فيما بعد في البرنامج .

## ١٠-١ التخصص التلقائي :

لنفترض أننا استعملنا متغيراً دون تحديد نوعه بواسطة جملة مبينة فما الذي يحدث باترى ؟ في الواقع أنه في هذه الحالة فإن الحاسب يقوم تلقائياً بتحديد نوع المتغير حسب القاعدة التالية :

١ - إذا كان اسم المتغير يبدأ بأحدى الحروف من A الى H أو من O الى Z فإنه يعامل معاملة متغير حقيقي ( عشري ) REAL .

٢ - إذا كان اسم المتغير يبدأ بأحد الحروف من I الى N فإنه يعامل وكأنه متغير صحيح INTEGER .

إذا الجملة المبينة ( التوضيحية ) غير لازمة بالضرورة ولكن من الأفضل بالتأكيد إستعمال الجمل المبينة ( التوضيحية ) وذلك للأسباب التالية :

١ - أن الجمل المبينة يحكم موقعها في بداية البرنامج تؤدي الى حصر جميع المتغيرات التي سوف تستعمل في البرنامج وهذا يؤدي بالتالي الى سهولة قراءة البرنامج من قبل الآخرين كما يعطي انطباعاً جيداً عن المبرمج من حيث التنظيم والبساطة .

٢ - استعمال الجمل المبينة يعطي المبرمج مرونة في استعمال الأسماء المناسبة والتي يمكن أن تصف المتغير الوصف المناسب الذي يعطي قارئ البرنامج صورة واضحة عن ماهية المتغير وعن الغرض المستعمل له .

مثال (٩) :

لنفترض أننا نريد أن نختار اسماً لمتغير يمثل عدد الأميال التي تقطعها سيارة في خلال ساعة واحدة فبالنسبة للمبرمج الفطن فإنه من البديهي أن يختار اسم MILES لهذا المتغير والذي يعطي صورة واضحة عن المتغير ونفترض أن عدد الأميال يمثل بأرقام عشرية من نوع REAL ، ففي هذه الحالة يجب علينا استعمال الجملة المبينة REAL MILES .

والتي ليس لنا غنى عنها إذا أردنا استعمال هذا الاسم ذو المحتوى العشري . أما اذا لم نرغب في استعمال جملة مبينة فإننا لانستطيع استعمال اسم MILES لأن الحاسب في هذه الحالة يعامل هذا

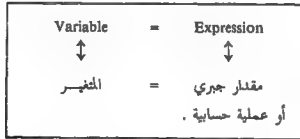
المتغير تلقائياً على أساس أنه رقم صحيح رغم أنه في الواقع رقم عشري . وبالتالي فإن هذا يؤدي الى استعمال الاسم الغير مناسب للمتغير المطلوب مما يجعل البرنامج أكثر صعوبة وأقل تنظيماً .

## ١٩-٢ جملة التعيين : ASSIGNMENT STATEMENT

تكلمنا فيما مضى عن تسمية متغير ما وعرفنا أن التسمية إنما تعني تسمية خلية بأسم معين ، اما نوعية محتواها فهو خاضع للجملة المبينة التي تحوي المتغير في قائمتها والا فإنه يخضع للتحديد التلقائي حسب القاعدة السالفة الذكر .

ولكننا لم نتحدث عن طريقة ادخال تلك القيمة في الخلية التي تحمل ذلك الاسم علماً أنه من الضروري جداً ادخال تلك البيانات في خلاياها الخاصة حتى يتمكن الحاسب من معالجتها عن طريق وحدة العمليات الحسابية والمنطقية التي تتصل بهذه الخلايا الموجودة في الذاكرة ومن ثم تُجرى عليها العمليات الحسابية والمنطقية المطلوبة في البرنامج .

وفي الواقع فإن هناك طريقتان مختلفتان لايصال القيم العددية الى خلاياها الخاصة بها وأحدى هاتين الطريقتين مباشرة والأخرى غير مباشرة ولنبدأ بالطريقة المباشرة البسيطة : وتتلخص هذه الطريقة في ايصال قيمة المتغير الى خلية عن طريق تعيين تلك القيمة بواسطة معادلة حسابية تسمى « جملة التعيين » Assignment Statement وتعتبر هذه الطريقة مباشرة لأنها توضع ضمن سطور البرنامج في الشكل العام التالي :



أما المتغير فقد تكلمنا عن تسميته سابقاً . يبقى لنا أن نتكلم عن المقدار الجبري وإشارة المساواة « = » . أما المقدار الجبري فهو المقدار الذي يكتب على يمين إشارة المساواة وقد يأتي في إحدى الصور التالية :

١ - مقدار ثابت - ومثال ذلك  $X = 1.0$

٢ - متغير واحد فقط ، ومثال ذلك  $X = Y$

٣ - مقدار جبري يشمل بعض المتغيرات والثوابت التي يفصل بينها اشارات العمليات الحسابية المختلفة من أسس وجمع وطرح وضرب وقسمة وبعض العمليات الأخرى ومثال ذلك

$$X = (2 + Y) / Z$$

أما العمليات الجبرية الأساسية فهي كالتالي :

المثال بلغة الفورتران	مثال رياضي	الإشارة في لغة الفورتران	الإشارة الرياضية	العملية الحسابية
A + B	a + b	+	+	الجمع
X - Y	x - y	-	-	الطرح
A * B	a x b	*	x	الضرب
A/B	a/b, $\frac{a}{b}$ , a : b	/	÷ و /	القسمة
X ** Y	(x) <sup>y</sup>	**		الأس

#### Order of Arithmetic Operations

٢-١ ترتيب العمليات الحسابية :

ياترى كيف تتم عملية ترتيب العمليات الحسابية بواسطة الحاسب الآلي وما هو النظام الذي يتبعه في هذا الشأن . لنأخذ مثلاً المقدار التالي :

$$5 * 3 + 2$$

لو كان ترتيب العمليات يتم من اليسار الى اليمين فأن عملية الضرب تتم أولاً لنحصل على (5\*3 = 15) ثم نضيف اليه 2 ليكون الناتج 17 . أما لو كان ترتيب العمليات معاكساً لهذا الافتراض ، أي أن ترتيب العمليات يتم من اليمين الى اليسار لحصلنا على (3 + 2 = 5) ثم ضربنا الناتج في 5 لنحصل على الناتج 25 . اذا من البديهي أن نستنتج أن أحد الجوابين غير صحيح وبالتالي فأن ترتيب العمليات الحسابية لابد أن يكون مهماً جداً بالنسبة لدارسي الفورتران - وفي الواقع أن الترتيب يتم كالتالي :

أولاً : الأقواس ( )

ثانياً : الأس \*\*

ثالثاً : الضرب والقسمة / و \*

رابعاً : الجمع والطرح + و -

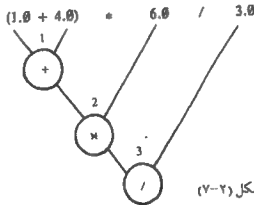
أما إذا تساوت عمليتين في الترتيب كعمليتي ضرب وقسمة أو عمليتي ضرب فإن التنفيذ يكون من الشمال الى اليمين .

مثال (١٠) :

$$(1.0 + 4.0) * 6.0 / 3.0$$

أوجد ناتج المقدار الجبري

الحل : حسب قواعد الترتيب التي ذكرناها نستنتج أن الترتيب يتم كالآتي :



شكل (٧-٧)

وبالتالي فالناتج هو  $3.0 / 6.0 * 5.0$  والذي يساوي  $10.0$  .

مثال (١١) :

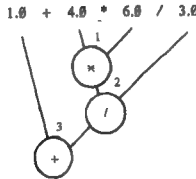
لاحظ أن هذا المقدار مشابه

$$1.0 + 4.0 * 6.0 / 3.0$$

أوجد قيمة المقدار الجبري

للمقدار السابق مع غياب القوس فقط .

الحل :



شكل (٨-٧)

من الواضح أن الناتج سيكون حسب الشكل أعلاه مساوياً

$$8.0 + 1.0 = (24.0 / 3.0) + 1.0$$

$$9.0 =$$

مثال (١٢) :

أوجد قيمة المقدار الجبري  $4.0 * (3.0 * (X + (Y - Z)) / (A + B)) - 5.0 * (A - B) ** 2$

ولنفترض أن لدينا القيم التالية :

$$X = 1.0$$

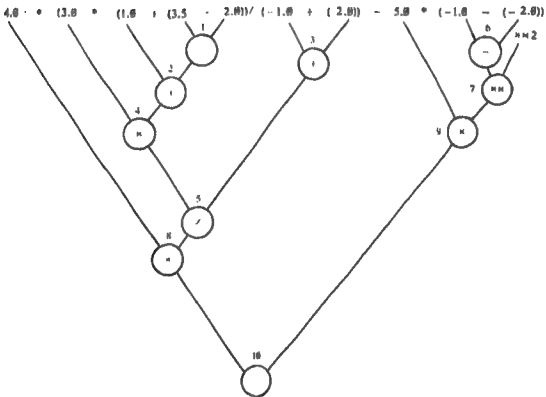
$$Y = 3.5$$

$$Z = 2.0$$

$$A = -1.0$$

$$B = -2.0$$

الحل :



شكل (١٢)

في الشكل أعلاه نستنتج أن المقدار سوف يصبح حسب ترتيب العمليات أعلاه :

$$4.0 * ((3.0 * 2.5) / (-3.0)) - 5.0 * (1.0) ** 2$$

والذي يساوي :

$$4.0 * (-2.5) - 5.0 * 1.0 = -10.0 - 5.0 = -15.0$$

ومن الملاحظ أنه إذا كانت هناك أقواس بداخل قوس فإن الحاسب يبدأ بتنفيذ العملية الحسابية في القوس الأصغر ثم الأكبر فالأكبر إلى حد سبعة أقواس بداخل بعضها البعض بالنسبة لمعظم الحواسيب الآلية . ومن الملاحظ أيضاً أن عدد الأقواس المفتوحة على الجهة اليمنى يجب أن يكون مساوياً لعدد الأقواس المفتوحة على الجهة اليسرى .

## ٢-١٣ ضرورة استعمال الأقواس :

$$\frac{a + b}{c + d} \text{ : نفترض أنه طلب منا كتابة المقدار الرياضي التالي :}$$

ولنفرض أننا كتبنا المقدار في الشكل التالي  $A + B / C + D$  دون أن نهم بوضع أي أقواس للبسط والمقام ، ولكن الواقع أن هذا المقدار يساوي  $A + (B/C) + D$  وهو مختلف تماماً عن المطلوب والذي يجب أن يكتب في الشكل  $(A + B) / (C + D)$  حتى يعطى المطلوب تماماً .

إذاً من الضروري الاستعانة بالأقواس لكتابة المقدار بالطريقة الصحيحة كما أن استعمال الأقواس وإن لم يكن ضرورياً في بعض الأحيان إلا أنه يساعد على توضيح المقدار الجبري وبخاصة إذا كان المقدار طويلاً ومعقداً .

مثال (١٣) :

المقدار  $A + B / C * D$  يساوي تماماً المقدار  $A + ((B / C) * D)$  .  
ولكن قراءة المقدار الثاني أسهل وأوضح ولا تدع مجالاً للتخمين إطلاقاً بالرغم من أن الأمر لا يحتاج إلى وضع أقواس على الإطلاق ولكن المبرمج الجيد يختار دائماً الطريق الواضح السهل على الآخرين سلوكه والاستفادة منه .

## ٢-١٤ اشارة المساواة :

بقي لنا بالنسبة لجمل التعيين أن نتكلم عن اشارة المساواة وما تعنيه بالنسبة للحاسب الآلي . ففي الرياضيات إذا كتبنا المعادلة التالية  $A = A + 1$  ثم حاولنا حل هذه المعادلة فأننا نحصل على الحل اللامعقول ٠ و ١ .

ولكن الأمر يختلف بالنسبة للحاسب الآلي والسبب يكمن في تفسير معنى إشارة المساواة بالنسبة للحاسب فهي لا تعني المساواة تماماً كما يعرفه الجميع في علم الحساب وإنما تعني التالي :

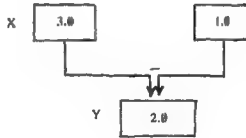
أوجد قيمة المقدار الجبري الموجود على يمين إشارة المساواة ثم ضع هذه القيمة الناتجة في الخلية التابعة للمتغير المكتوب على يسار إشارة المساواة . أما إذا كانت هناك قيمة سابقة للمتغير فإن الحاسب يقوم باستبدال القيمة السابقة بالقيمة الحالية الناتجة .

مثال (١٤) :

أفرض أن  $Y = X - 1.0$  وأن قيمة  $X$  تساوي  $3.0$  ، أوجد قيمة  $Y$  ؟

الحل :

لتصور أن كلاً من  $X$  و  $Y$  يمكن تمثيله بخلية على شكل مستطيل كما في الشكل (١٠-٢) وكما يظهر في الشكل أن قيمة  $Y$  تساوي  $2.0$



شكل (١٠-٢)

مثال (١٥) :

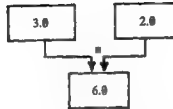
أفرض أن قيمة  $X$  الحالية تساوي  $3.0$  أوجد قيمة  $X$  من المعادلة الحسابية :

$$X = X * 2.0$$

الحل : من الشكل (١١-٢) يتضح أن قيمة  $X$  الجديدة تساوي  $6.0$

وأن هذه القيمة تم عمل القيمة القديمة لـ  $X$  والتي تساوي  $3.0$  والتي لا يمكن استرجاعها حيث أنها فقدت تماماً بعد عملية استبدالها بالقيمة الجديدة .





شكل (١١-٢)

أنظر الى السطور التالية والتي تمثل جزءاً من برنامج :

```

REAL X, Y, Z
  X = 10.0
  Z = X + Y
  Y = 5.0
  
```

في البرنامج الموضح أعلاه ، عندما يصل الحاسب الى السطر الثالث فإنه سيقوم باستعمال أي قيمة سابقة موجودة في الخلية Y لحساب قيمة Z على الرغم من أن ذلك ليس هو الوارد في ذهن المبرمج . ولاحظ أيضاً أن الحاسب لن يتعرض على تنفيذ هذا السطر .

```

REAL X, Y, Z
  X = 10.0
  Y = 5.0
  Z = X + Y
  
```

أي أنه عند تنفيذ أي عملية حسابية يجب أن تكون قيم جميع المتغيرات المستعملة في الطرف الأيمن من إشارة المساواة معلومة حتى يمكن أنيجاد قيمة المتغير المكتوب في الطرف الأيسر من المعادلة .

## «تساوين عامة»

١ - في نهاية الجزء التالي من برنامج ما ، ماهي قيم كل من المتغيرين A و B ؟

INTEGER A , B

$$B = 10$$

$$A = B$$

$$B = 2$$

٢ - ماهي القيم التي ستعطى للمتغير الصحيح B في كل من جمل التعيين التالية :

$$B = 2*3*4 / 4$$

$$B = (2/1) + 1$$

$$B = -18*3$$

$$B = -4 * 5 + 2$$

٣ - أي من التالي يُعتبر جملة تعيين شرعية في لغة الفورتران ؟ علل في حال النفي ! ( افترض أن جميع المتغيرات تمثل أرقاماً صحيحة ) .

$$A = A*A + A$$

$$BAKR = 3$$

$$MILE + FEET = 45$$

$$CAT * DOG = - 16 + CAT$$

$$HAMID = HASAN + FAHD$$

$$LAI + JAMAL - 5$$

$$3BOB = 476$$

$$ALI = 3(15) + AHMED$$

$$RICE = 16 + RICE$$

$$OLIVE = FRTRAN / 4986132$$

$$NO + YES = ORDER$$

٤ - ماهي القيم التي ستعطى للمتغير الصحيح FLOOS في كل من جمل التعيين التالية ؟

$$FLOOS = 2 + (8*3) / 4$$

$$\begin{aligned} \text{FLOOS} &= 323 / 2 \\ \text{FLOOS} &= (27 / 28) + 3 \\ \text{FLOOS} &= (21 / 20) - 1 \\ \text{FLOOS} &= (8/16) * 1000 \end{aligned}$$

٥ - اكتب جملاً بلغة الفورتران تقوم بالمهام التالية :

- ( أ ) تبين أن أحد خلايا الذاكرة سيطلق عليها اسم ZAID وستعطى القيمة الصحيحة 2 .  
 (ب) تعين لـ ZAID قيمته السابقة مضروبة في نفسها ٥ مرات .  
 (جـ) تعين لـ ZAID قيمته السابقة بالإضافة الى Z .

٦ - أي من الأرقام التالية يعتبر « ثابت صحيح » ؟ علل في حالة النفي !

1	3 * 4
1.0	36.7
-13	-129.4
-.987654321	376
60 + 0	-0

٧ - أي من الأرقام التالية يعتبر « ثابت حقيقي » ؟ علل في حالة النفي ؟

1	5.86
2.0	5.68 + 3
-3.00	-.00001
-4.4 + 4.0	367
-3.0 * 2	-15. + 4.0

٨ - اكتب جملة « فورتران » مساوية لكل جملة من الجمل الرياضية التالية :

- |                      |                           |
|----------------------|---------------------------|
| 1. $X + y^2$         | 6. $a - \frac{bc}{a + d}$ |
| 2. $(X + y)^3$       | 7. $2x - 6y$              |
| 3. $X^4$             | 8. $x/y^2$                |
| 4. $a + (b \div c)$  | 9. $(x+y)^2 / x^2 - y^2$  |
| 5. $\frac{a + b}{c}$ | 10. $\frac{ay}{3d} - bx$  |

٩ - أوجد قيمة المقادير التالية :

$$\begin{aligned} & -1.0^{**4} \\ & 3.0^{*3.**2} \\ & 4/5 * 2 + 3.0 \\ & -3^{**2**3} \\ & (6.0 - 3.) / 4.0^{**2} \\ & 2.*3.0 - 7./14.0 + 6.2 \\ & (2.0)^{**2.5} \end{aligned}$$

١٠ - لنفرض أن  $A = -1.0$  علل لماذا تكون  $A^{**2}$  صحيحة بينما لا تعتبر  $A^{**2.0}$  شرعية في لغة الفورتران ؟

١١ - استعمل الأقواس لتوضيح المقادير الجبرية التالية :

$$ALI + HAMID * FAHD ** 2$$

$$METER + CM - FEET * TIME + KM ** POWER ** DIST \\ L/N/M$$

١٢ - في البرنامج التالي تتغير قيمة المتغير RESULT عدة مرات أثناء سير البرنامج . أوجد جميع هذه القيم ؟

```
INTEGER A,B,C
REAL RESULT , R1 , R2
R1 = 1
R2 = 2
A = 1
B = 4
C = 16
RESULT = R1 * R2
RESULT = A * R2
RESULT = B/(A*B)
RESULT = B - C
RESULT = - C/B
STOP
END
```

١٣ - عند كتابة مقدار جبري يمكننا أن نستعمل الرمز  $ab$  لنعني حاصل ضرب  $a$  و  $b$  بينما لا يجوز ذلك في لغة الفورتران . علل ذلك ؟

## الفصل الثالث

### تعليمات إدخال وإخراج البيانات



## الفصل الثالث

### Input / Output Statements

### تعليمات إدخال وإخراج البيانات

مقدمة : ذكرنا فيما مضى أن جملة التعيين التي تأخذ الشكل العام .

Variable = expression

المقدار الجبري = المتغير

تعطى للمتغير القيمة الناتجة من حساب قيمة المقدار الجبري ، وحيث أنه من الممكن أن يكون المقدار الجبري مقداراً ثابتاً constant فإنه من الواضح أن جملة التعيين التي تأخذ الشكل العام .

Variable = Constant

ثابت = المتغير

يمكن اعتبارها وسيلة مباشرة من وسائل إدخال البيانات الى الحاسب الآلي ، فمثلاً لو نظرنا الى الجزء التالي من برنامج ما :

REAL X, Y, SUM

X = 2.0

Y = -1.0

SUM = X + Y

شكل (١-٣)

فانه من الواضح أن قيمتي X و Y قد زودت للحاسب عن طريق جملي التعيين في السطرين الثاني والثالث ، وبالتالي فإن الحاسب يستطيع أن يزودنا بقيمة SUM والتي تساوي مجموع قيمتي X و Y كما هو موضح في السطر الرابع من البرنامج ، وهكذا لا بد لنا أن نعطي بعض قيم المتغيرات لكي نحصل على قيم جديدة نحتاجها .

فالعملية اذن عبارة عن أخذ وعطاء ولهذا فإن عملية إدخال البيانات يعتبر جزءاً أساسياً وعنصراً هاماً في مجال البرمجة الالكترونية كما هو واضح في المثال السابق .

لنفترض الآن أن بين أيدينا بيانا يشتمل على ألف قيمة لا بد لنا أن نغذيها للحاسب الآلي حتى نحصل على بعض النتائج المطلوبة كحاصل جمعها أو ضربها أو المعدل أو ماشابه ذلك . من البديهي أن يفكر القارئ المبتدئ في استعمال ألف جملة تعيين مختلفة ليزود الحاسب بالقيم الموجودة في

البيان ، ولكن من الواضح أن هذه الفكرة غير منطقية وغير عملية على الإطلاق لأنها في الحقيقة تفقد الحاسب الآلي أهم خصائصه على الإطلاق ألا وهي عملية اختصار الوقت والجهد ، لذا كان ولا بد من وجود وسائل أخرى أكثر سرعة وبساطة وسهولة لادخال البيانات الى الحاسب بدون اضافة الوقت والجهد في كتابة جمل تعين بعدد القيم الموجودة في البيان .

### The READ Statement

٣-١ جملة القراءة

وتعتبر هذه الجملة من أهم الوسائل الغير مباشرة لادخال البيانات الى الحاسب الآلي لأن الجملة نفسها تكون ضمن نطاق البرنامج الفعلي أما البيان فيمكن التفكير فيه كملحق للبرنامج لا يكون ضمن حدود البرنامج الفعلية .

البرنامج الفصل	:	البيان المرفق
	READ Statement	
	:	
	END	
	:	

شكل (٣-٢)

أما الشكل العام لجملة اقرأ فهو كالتالي :

READ (n, m) V1, V2, V3, V4, .....

حيث n هو رقم الجهاز القارئ وعادة ما يكون 2 أو 5 .

m هو رقم الجملة الشارحة :

V1, V2, V3, ... قائمة بأسماء المتغيرات المطلوب قراءة قيمها من البيان Input Data حسب تسلسلها في القائمة .

ولنوضح قليلاً كل عنصر من عناصر جملة اقرأ . فالرقم n عادة ما يكون رقماً ثابتاً صحيحاً تحدد نوعية الجهاز المستعمل للقراءة ( انظر ١-٢ ) ، فهناك قارئ للبطاقات «Card Reader» وهناك قارئ للاشرطة المغنطة «Tape drive» وفي بعض الأحيان تكون القراءة عن طريق الآلات الكتابية عن بعد «Teletypes» أو عن طريق الشاشات والنوع الأخير هو النوع السائد الاستعمال حالياً



خصوصاً للمبتدئين . أما الرقم m فهو رقم جملة شارحة تشرح الطريقة التي كتبت بها القيم في البيان المرفق . فمن الجائز أن نكتب أربع قيم في السطر الأول وقيمة واحدة فقط في السطر الثاني وقد يختلف الوضع فتكتب قيمتين في السطر الأول وثلاث قيم في السطر الثاني ... وهكذا بالنسبة لبقية الأسطر في البيان ، لذا فإن وجود الجملة الشارحة ضروري وهي جزء من البرنامج وستكلم عنها بالتفصيل فيما بعد .

أما قائمة المتغيرات فهي أما أن تتكون من متغير واحد ، أو أكثر من متغير مفصولة بفواصل «Comma» أما وظيفة جملة اقرأ فهي أنها تأمر الحاسب بقراءة القيم الموجودة في البيان المرفق طبقاً لوصف القراءة الذي تشترطه الجملة الشارحة ثم وضع هذه القيم في خلايا تحمل الأسماء الموجودة في القائمة التي تعتبر جزءاً من الجملة نفسها . أما كيفية كتابة هذه القيم وترتيبها في البيان المرفق فهذا يعتمد تماماً على الجملة الشارحة التي تحمل الرقم m .

مثال (١) :

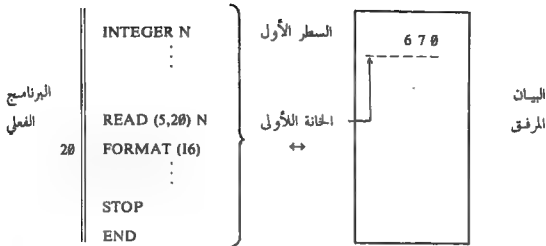
READ (5 , 20) A, B,MAN	(أ)
READ (5 , 300) METER, MILE	(ب)
READ (5 , 6441) X	(جـ)

### ٣-٢ وصف البيانات :

لكل نوع من أنواع القيم ، يوجد نوع خاص من واصفة البيانات «data descriptor» فالتعريف الذي يصف الأعداد الصحيحة «INTEGERS» يختلف عن التعريف الذي يصف الأعداد الحقيقية «REALS» فمثلاً الوصف «IW» يُستعمل لوصف الأعداد الصحيحة أي أن 16 يستعمل للدلالة على عدد صحيح في البيان يشغل ست خانات في السطر المقروء ، أما كيف تم هذه الدلالة فالأمر يكون واضحاً إذا ما نظرنا إلى المثال التالي :

20		READ (5 , 20) N
		FORMAT (I6)

فالجملة الأولى تأمر الحاسب بأن يقرأ قيمة N من البيان المرفق حسب الجملة الشارحة ذات الرقم 20 وعندئذ يقوم الجهاز القارئ للبيانات بقراءة قيمة N من خلال الست خانات الأولى في السطر الأول من البيان . أما الآن فلنمثل هذه العملية بيانياً حتى تتضح الصورة في ذهن القارئ :



الشكل (٣-٣)

ومما يلاحظ أن قيمة  $N$  المساوية لـ 670 لا تشغل سوى ثلاث خانات رغم أننا قد حجزنا لها عن طريق الجملة الشارحة ستة خانات ، ولكن هذا لا خلاف فيه طالما أن قيمة  $N$  قد كتبت في الوضع الصحيح . ونقصد بالموضع الصحيح ملاءم الخانات التي في أقصى اليمين أولاً بأول ، فمثلاً لو كتبنا قيمة  $N$  على الشكل التالي :

- 6 7 0 - -  
 الخانة الأولى .

لاعتبرها الحاسب كما لو أنها 67000 أي أنه سوف يملأ الخانات التي في أقصى اليمين بأصفار . إذا لابد لنا أن نراعي هذه النقطة وأن نكون حذرين جداً عند كتابة قيم المتغيرات في البيان المرفق كما لابد لنا أن نراعي أن العدد السالب يجب أن تسبقه إشارة «-» والتي تأخذ إحدى الخانات المحجوزة لقيمة المتغير .

أما في حالة المتغيرات الحقيقية فإن واصف البيان يكون على الشكل Fw. d حيث  $w$  يساوي مجموع الخانات المحجوزة للعدد الحقيقي بينما  $d$  تساوي مجموع الخانات المحجوزة للجزء العشري من الرقم كما تحجز خانة واحدة للنقطة العشرية . وبالتالي فإن عدد الخانات المحجوزة للجزء الصحيح من الرقم تساوي  $w-d-1$  .

مثال :

اكتب الرقم الحقيقي 31.45- حسب واصفه البيان F7.2 .

الحل :

مجموع الخانات المحجوزة للرقم هي ٧ خانات ، اثنتين منها للجزء العشري وواحدة للنقطة العشرية ، وبالتالي تبقى ٧-٢-١=٤ خانات للجزء الصحيح الذي يتضمن الإشارة السالبة ( أنظر الشكل (٤-٣) ) .

$$\begin{array}{r} -31.45 \\ \hline \text{---} \end{array}$$

w-d-1      d

الشكل (٤-٣)

إذا مما سبق نستنتج أن Iw تكون للأعداد الصحيحة بينما Fw.d تكون للأعداد الحقيقية .

مثال : لنفترض أن لدينا الجزء التالي من برنامج ما .

البرنامج		REAL XSUM	4 0 . 3 2 - 1 5
		INTEGER NUMBER	--- -- -- --
		READ (5,1) XSUM, NUMBER	
الفعلي	1	FORMAT (F 5.2,I3)	
		:	
		STOP	
		END	

البيان المرفق

الشكل (٥-٣)

لاحظ أن قيمة XSUM شغلت جميع الخانات المخصصة لها وهي ٥ خانات وكذلك الحال بالنسبة لقيمة NUMBER ولكن ماذا سيحدث لو أن قيمة XSUM كانت فعلاً 40.32 - فأنا عندئذ يجب علينا أن نغير من واصف البيان F5.2 الى F6.2 لأن الواصف السابق لا يكفي للقيمة التي نحتاج الى ستة خانات على الأقل . اذاً لا بد لنا من مراعاة حجم القيم المستخدمة في البيان وبالتالي نستخدم واصف البيان الذي من الممكن أن يغطي جميع القيم الموجودة في البيان ، كأن تستعمل في هذا المثال F10.2 أو أن نستخدم I6 بدلاً من I3 وهكذا مع مراعاة التقيد بالأنظمة المذكورة آنفاً من حيث كتابة البيان طبقاً للنصوص واصفات البيان في الجملة الشارحة .

### The WRITE Statement

٣-٣ جملة الطباعة :

تعلمنا حتى الآن كيف نطلق الأسماء على خلايا الذاكرة وتعلمنا كيف نعطي هذه الخلايا القيم الخاصة بها عن طريق جملة التعيين أو عن طريق جملة اقرأ ، وتعلمنا كيف نوجه الأوامر الى الحاسب ليؤدي بعض المهام الحسابية والجبرية - ولكننا لم نتعلم حتى الآن كيف نأمر الحاسب بطباعة النتائج المطلوبة ليتسنى لنا الاستفادة منها لأنه من الجائز أن يكون الحاسب قد قام بإيجاد قيمة متغير ما وحفظ تلك القيمة في الخلية الخاصة بالمتغير ، ولكن حتى يمكننا الاستفادة من تلك النتيجة - يجب علينا أن نحصل على تلك القيمة مكتوبة أمامنا ، والا فأن إيجاد تلك القيمة أو عدمه يتعب متساويان في نظر المبرمج ولكانت كتابة البرامج مضنية للوقت والجهد والمال .

مثال : انظر الى البرنامج التالي :

```
REAL X, Y, RESULT
X = 1.56
Y = -29.54
RESULT = Y**2 + X*Y
STOP
END
```

الشكل (٣-٦)

اذا ما نظرنا الى هذا البرنامج لأول وهله ، فأننا قد نعتقد أن البرنامج قام بإيجاد قيمة المتغير RESULT وأننا قد حصلنا على المطلوب . ولكن الواقع أن الحاسب يكون قد أوجد فعلاً - بعد تنفيذ البرنامج - قيمة المتغير RESULT ومن المستحيل بالنسبة لنا أن نطلع على خلايا الذاكرة لنحصل على هذه القيمة وعندئذ ندرك أن تنفيذ هذا البرنامج وعدم تنفيذه سيان بالنسبة لنا .

لذلك كان من البديهي أن تكون هناك تعليمة توجه الحاسب الى كتابة أو طباعة القيم والنتائج التي يرغب المبرمج في الحصول عليها بعد تنفيذ برنامج معين . أما الوسيلة لتحقيق ذلك فهو استعمال جملة اكتب ولكي ندرك وظيفة هذه التعليمة بوضوح ونفهم ماهيتها فهما شاملا ، يتوجب علينا أن ندرك الأجزاء المختلفة من هذه التعليمة .

في البداية علينا أن ندرك ( كما هو الحال بالنسبة لجملة اقرأ ) أن هناك أجهزة مختلفة يمكن ايصالها بالحاسب لتقوم بمهمة الكتابة ( انظر ٢-١ ) ، ومن هذه الأجهزة الطابعات الخطية Line Printers والأشرطة المغنطية «Magnetic Tapes» وغير ذلك من الأجهزة الا أن الجهاز الشائع الاستعمال بالنسبة للطالب المبتدئ هو الطابعات الخطية أو الوحدات الطرفية «Teletypes» .

ولكي يصبح بالأمكان للحاسب أن ينفذ تعليمة اكتب فإنه وكما هو الحال بالنسبة لتعليمة اقرأ يجب علينا أن نحدد نوع الجهاز المستعمل للكتابة كما يجب علينا أن نحدد الجملة الشارحة **FORMAT Statement** وعلينا أن نوضح أيضاً أسماء المتغيرات المطلوبة كتابة قيمها . لذا فإن الشكل العام لجملة « اكتب » هو التالي :

**WRITE (n , m) V1, V2, V3, ...**

حيث **n** هو رقم الجهاز المستعمل للكتابة وغالباً ما يكون «6»

**m** هو رقم الجملة الشارحة **FORMAT Statement number**

**V1, V2, V3, ...** قائمة بأسماء المتغيرات المطلوب كتابة قيمها حسب تسلسلها في القائمة .

أما المعنى العام فهو كالتالي : استعمل الجهاز ذو الرقم **n** لكتابة قيم المتغيرات **V1, V2, V3, ...** بالتسلسل وحسب الوصف المرفق في الجملة الشارحة ذات الرقم **m** .

أمثلة :

**WRITE (6,1036) A,B,CX**

**WRITE (6,713) SUM**

**WRITE (6,204) X1, X2**

**WRITE (6,593)**

أما الآن فنتكلم قليلاً عن الجملة الشارحة **FORMAT Statement** بقليل من التفصيل ، ووظيفة الجملة الشارحة هي شرح طريقة كتابة النتائج المطلوبة سطرًا سطرًا ، وهناك عدة طرق لتحديد موقع السطر التالي من البيانات الناتجة ، أما أهمها فهي الطرق التالية :

أولاً : أن يكتب السطر التالي بعد السطر الحالي مباشرة .

ثانياً : أن يترك سطر خال بعد السطر الحالي ثم يكتب السطر التالي .

ثالثاً : أن يكتب السطر التالي في بداية الصفحة التالية مع ترك باقي الصفحة الحالية خالياً .

ويم تحديد أي من هذه الاختيارات باستخدام ما يسمى برمز التحكم الحركي **«Carriage control character»** والتي تكتب ضمن الجملة الشارحة بين قوسين صغيرين وكما هو موضح في الشكل (٣-٧) .

' ( فراغ واحد ) : يعني تحرك الى السطر التالي في نفس الصفحة قبل أن تبدأ الطباعة .

' 0 ( صفر ) : يعني اترك السطر التالي خالياً ثم أبدأ الطباعة في السطر الذي يليه .

' 1 ( واحد ) : يعني اترك بقية الصفحة الحالية خالية ثم أبدأ الطباعة في السطر الأول من الصفحة التالية .

الشكل (٧-٣)

أما موقع هذا الرمز فيجب أن يكون في بداية الجملة الشارحة لكي تشير الجهاز الطابع بموقع السطر التالي طبقاً لرغبة المبرمج .

مثال :

```
251 | WRITE (6,251) X, Y
    | FORMAT ('0', F4.1, F5.2)
```

من النظرة الأولى ندرك أن المطلوب هو طباعة قيمتي  $X$  و  $Y$  حسب الجملة الشارحة ذات الرقم 251 . وبمجرد أن يجد الجهاز الطابع الرمز '0' فإنه يدرك أن عليه أن يترك سطرًا خاليًا ثم يبدأ الكتابة في السطر الذي يليه حيث يقوم بطباعة قيمتي  $X$  و  $Y$  حسب واصفات البيان الموضحة .

والآن لنفترض أننا نرغب من الحاسب أن يكتب جملة كتابة حرفية دون تغيير أو تبديل ، وذلك بقصد توضيح وتسهيل قراءة نتائج البرنامج «Program Output» أو لأي غرض آخر ، فعندئذ كل ما نحتاجه هو وضع الجملة المطلوب طباعتها حرفياً بين أقواس صغيرة Quote marks كما فعلنا سابقاً بالنسبة لرموز التحكم الحركي ( انظر الشكل ٧-٣ ) .

مثال : انظر الى البرنامج التالي :

```
100 | X = 456.14
    | Y = -310.52
    | SUM = X + Y
    | WRITE (6, 100) X, Y, SUM
    | FORMAT (' ', 'THE SUM OF', F7.2, 'AND', F8.2, 'IS', F7.2)
    | STOP
    | END
```

أما الناتج الذي سوف نحصل عليه فسيكون كالتالي :

الخاصة الأولى

↑ THE SUM OF B 456.14 B AND B - 310.52 B IS B 145.62

حيث B تعني blank أو خانة فارغة . وبهذا يتضح لنا أن استعمال الواصفات الحرفية «Literal descriptors» تجعل قراءة الناتج أكثر وضوحاً للقارئ الذي يطلع على نتائج البرنامج للاستفادة منها رغم أن المبرمج لا يجد صعوبة في قراءة الناتج وذلك لأنه خبير بالبرنامج مطلع على كل صغيرة وكبيرة فيه ، ولكن كما أسلفنا أن المبرمج يكتب البرنامج ليستفيد هو منه شخصياً كما يستفيد منه الآخرون بنفس المقدار لهذا فلو أن البرنامج السابق كتب بنفس الطريقة السابقة مع استبدال الجملة الشارحة رقم 100 بالجملة :

100 FORMAT (F7.2, F8.2, F7.2)

لكان الناتج على الشكل التالي :

الخاصة الأولى

↑ 456.14 B - 31.52 B 145.62

والذي تصعب قراءته على من يقرأ الناتج لأول مرة رغم بساطة البرنامج ورغم أن النتيجة هي نفس النتيجة السابقة فما بالك عندما يكون البرنامج متشابهاً ويحتوي على عشرات الأسطر من النتائج المختلفة التي تحتاج الى إيضاح وتبسيط .

إذا الواصفات الحرفية Literal descriptors لا تغير شيئاً من النتائج ولكنها تؤدي دوراً كبيراً في سبيل تبسيط قراءة نتائج البرنامج حتى على المبرمج نفسه ، وخاصة عندما يعود الى قراءة البرنامج بعد زمن طويل من كتابته واستعماله .

فيما سبق استعرضنا بعض الجمل الشارحة وكيفية استخدامها في إدخال البيانات أو إستخراج النتائج ، ولكن من المعروف أن لغة الفورتران تحوي الكثير من الامكانيات التي يمكن استخدامها في تنظيم إدخال البيانات أو كتابة النتائج بالصورة التي يراود الحصول عليها . وفيما يلي سنستعرض بإستخدام أمثلة مختلفة ، بعض الجمل الشارحة الأخرى التي تستخدم في لغة الفورتران والتي منها :

٣-٤ استخدام الحرف A في الجملة الشارحة :

يستخدم الحرف A في الجملة الشارحة اذا ما أريد قراءة أو طباعة بيانات تحتوي على حروف أو أرقام أو كليهما معاً ، ولنا يرمز الحرف A الى كلمة Alphameric . وتم عملية تخزين الحروف والأرقام بإستخدام الحرف في عدد صحيح من الحلايا ، أي تم عملية التخزين بتقسيم البيان الى عدد

متساو من الحروف والأرقام ، وكل مجموعة تخزن في خلية واحدة . ومعظم الحاسبات لا تقبل أكثر من أربعة أرقام وحروف في كل خلية ، ومع ذلك إذا كانت مجموعة الحروف والأرقام التي يراد تخزينها في الخلية الواحدة أكبر من سعة تلك الخلية ، فأنها لن تقبل سوى مجموعة الحروف والأرقام التي في أقصى اليمين . أما إذا حدث العكس وكانت عدد الحروف والأرقام أقل من سعة الخلية فأن مجموعة الحروف والأرقام تلك تخزن في أقصى همال الخلية ويتم استكمال باقي الخلية بأصفار . فعل سبيل المثال إذا كانت كل خلية تتسع لأربعة حروف وأرقام فأنه إذا كان :

البيان المراد قراءته ( تخزينه )	واصف البيان	البيان المخزون
HODA	A4	HODA
SARWAT	A6	RWAT
OK	A2	OK

وعند إستخراج بيانات سبق تخزينها ، بأفترض أيضا أن كل خلية ستحتوي على عدد من الحروف والأرقام لا يزيد عن أربعة ، فأنه إذا كان :

البيان المراد إستخراجه ( طباعته )	واصف البيان	البيان المستخرج ( المطبوع )
KING	A4	KING
KING	A2	KING
KING	A6	BOOKING

مثال ١ :

إذا أريد إدخال ( قراءة ) البيان FACULTY OF SCIENCE فإنه يتم تقسيم ذلك البيان الى مجموعات من الحروف ، كل مجموعة لاتزيد عن أربعة حروف وبالتالي تكون تعليمية القراءة كالتالي :

9		READ (5,9) NA1, NA2, NA3, NA4, NA5
		FORMAT (3A4)

نلاحظ في هذا المثال أننا قسمنا البيان الى خمسة أقسام كل منها لا يزيد عن أربعة حروف . أي أن البيان سيشتغل على الأقل خمس خلايا مختلفة ، ونظراً لأن الخلايا مختلفة لذا أعطينا اسماً لكل منها وهي :

NA5 , ... , NA2 , NA1



مسال ٢ :

سبق تخزين البيان :

KINGABDABAZIZHUNIV.

في المتغيرات V1 , V2 , ... , V25 بحيث أن كل حرف كان قد سبق تخزينه في خلية واحدة .  
اكتب جملة الطباعة التي يمكنها تحقيق ذلك :

4	WRITE (6,4) V1, V2, ... , V25 FORMAT (IX, 25A1)
---	--

٣-٥ استخدام الحرف E في الجملة الشارحة :

عادة ما يستخدم الحرف E في الجملة الشارحة ، عندما تكون النتائج المراد إستخراجها صغيرة جداً أو كبيرة جداً . فعلى سبيل المثال إذا كانت قيمة المتغير A التي تم حسابها وتخزينها في الحاسب هي 0.000008 . فإذا أردنا طباعة قيمة المتغير A بالصيغة F6.4 مثلاً فإن النتيجة ستكون كالتالي :  
(0.0000) . كذلك إذا كانت قيمة A هي 5763.426 وأردنا طباعة تلك القيمة بنفس الصيغة السابقة ، فإن النتيجة ستكون كالتالي(\*) :

\*\*\*\*\* . وهذا يعني أن عدد الأرقام في الجزء الصحيح من النتيجة أكثر من العدد المراد إستخراجه بصيغة الطباعة ، وعلاج مثل تلك الحالات بسيط إذا كانت قيم المتغيرات في الحدود المعقولة والتي يمكن قراءتها كأن نكتفي بطباعة ثمانية أرقام عشرية بجانب القيمة الصحيحة للمتغير والتي قد تكون عشرة أرقام أخرى أو أقل . وفي مثل تلك الأحوال يمكننا تجنب الأخطاء التي قد تحدث نتيجة استخدام صيغة طباعة غير كافية بإستخدام صيغة طباعة كبيرة نسبياً مثل F20.8 مثلاً . ولكن في بعض المشاكل قد تكون قيم المتغيرات التي يتم حسابها داخل الحاسب كبيرة جداً مثل حسابات المشاكل الفلكية ، وقد تكون تلك القيم صغيرة جداً مثلما يحدث في المشاكل الفيزيائية أو الكيميائية والتي قد تأخذ قيم المتغيرات فيها القيمة ١٠<sup>-٢٠</sup> أو أقل من ذلك . وفي جميع مثل تلك الحالات ( إذا كان هناك تخوف من أن النتيجة قد تكون صغيرة أو كبيرة أو يصعب تقديرها ) ، فإننا نستخدم صيغة الأس Exponent لطباعة تلك القيم والتي يرمز لها بالرمز E في الجملة الشارحة . فالقيم التالية يمكن كتابتها بصيغة الأس كالتالي :

(هـ) بعض الحاسبات تعطي خطأ في مثل تلك الحالات .

$$\begin{aligned}
0.000008 &= 8 \times 10^{-6} = 8.0E-06 = 8.0E-6 \\
&= 0.8 \times 10^{-5} = 0.8E-05 = 0.8E-5 \\
&= 0.08 \times 10^{-4} = 0.08E-04 = 0.08E-4 \\
&= \text{-----} \\
&= 0.000008 \times 10^{-1} = 0.000008E-01 = 0.000008E-1 \\
&= 0.000008 \times 10^0 = 0.000008E00
\end{aligned}$$

كذلك فإن العدد :

$$\begin{aligned}
9876543.21 &= 9.87654321 \times 10^6 = 9.87654321E06 \\
&= 98.7654321 \times 10^5 = 98.7654321E05 \\
&= 987.654321 \times 10^4 = 987.654321E04 \\
&= \text{-----} \\
&= 9876543.21 \times 10^0 = 9876543.21E00 \\
&= 98765432.1 \times 10^{-1} = 98765432.1E-01 \\
&= 987654321.0 \times 10^{-2} = 987654321.0E-02
\end{aligned}$$

- من المثالين السابقين يمكن القول بأنه لقراءة أي عدد حقيقي في صورة أسية فإن العدد يتكون من جزئين يفصل بينهما الحرف E .

- الجزء الأول مجموعة من الأرقام ( حسب درجة التقريب المراد الحصول عليها ) وتحوي فيما بينها العلامة العشرية ومجموعة هذه الأرقام قد تكون موجبة أو سالبة وتسبق الحرف E .

- الجزء الثاني ويسمى بالأس Exponent ويكون عامة في الصورة  $XX \pm E$  وفي حالة ما إذا كان الأس موجياً فإن الطابع يهمل كتابة الإشارة ، بينما يكتب الإشارة (-) في حالة ما إذا كان الأس سالباً . وتختلف القيمة العظمى للأس مابين حاسب وآخر ، ففي الحاسبات IBM 1130 أو UNIVAC 1108 فإن قيمة الأس لن تزيد عن 38 بينما في الحاسبات IBM 360/370 لا تزيد عن 75 .

ولاعطاء أي قيمة من القيم السابقة الى الحاسب لقراءتها ، فإن صيغة القراءة باستخدام الحرف E تشبه الى حد كبير الصيغة باستخدام الحرف F .

مثال : اذا كانت :

$$A = -1.2685342 , B = 225.346$$

فإن القيم السابقة يمكن كتابتها باستخدام صيغة الأس E كالتالي :

المتغير	القيمة	القيمة بصورة F	القيمة في صورة أسية	القيمة باستخدام E
A	- 1.2685342	F10.7	- 1.2685342E + 00	E14.7
			- 12.685342E - 01	E14.6
			- 126.85342E - 02	E14.5
			⋮	⋮
			- 0.12685342E + 01	E15.8
			- 0.012685342E + 02	E16.9
			⋮	⋮
B	225.346	F 7.3	225.346E + 00	E11.3
			22.5346E + 01	E11.4
			⋮	⋮
			2253.46E - 01	E11.2
			22534.6E - 02	E11.1
			225346.0E - 03	E12.1
			000225346.0E - 03	E15.1

ويتضح من هذا المثال أنه لقراءة قيمة المتغير بالصورة E فإنه يقرأ في الصورة العامة Ew. d حيث w تمثل عدد جميع الأرقام التي يحتوي عليها العدد بما فيها حرف E وإشارتي العدد والأس والعلامة العشرية ، أما d فتمثل عدد الأرقام التي تتلو العلامة العشرية . فمثلاً

$$\begin{array}{c} d \\ \underbrace{-126.85342E - 02}_w \end{array}$$

أما عند طباعة قيمة المتغير المخزونة داخل الحاسب ، فإن هناك اختلافاً بسيطاً يحدث وهو أن الحاسب يقوم بتخزين العدد وطابعته بحيث أن العلامة العشرية تسبق أول رقم معنوي في العدد ( أي عدد يختلف عن الصفر ) ، بينما تتحدد قيمة العدد بقيمة الأس . فمثلاً العدد  $-126.85432E - 02$  يمكن طباعته بالحاسب بصيغ مختلفة منها :

صيغة الكتابة	صورة العدد بالصيغة المطلوبة
E14.8	- .12685342E + 01
E12.3	0000 - .127E + 01
E10.4	- .1269E + 01
E 9.2	0 - .13E + 01
E20.12	00 - .1268534200000E01
E 7.2	*****

في صيغة الكتابة الأخيرة لن تطبع قيمة المتغير نظراً لأن صيغة الكتابة لهذا المتغير تحتوي على :-

- الأس وإشارته وحرف E ( ٤ وحدات )
- رقمين عشريين ( ٢ وحدة )
- العلامة العشرية ( ١ وحدة )

و تبقى إشارة العدد نفسه التي لم يعمل حساب لها نظراً لأن صيغة الكتابة غير كافية . ولذا عندما يراد طباعة قيمة أي متغير يجب الأخذ في الاعتبار : تخصيص ٤ وحدات للأس وإشارته وحرف E بالإضافة الى تخصيص وحدة للعلامة العشرية ، وكذلك وحدة لإشارة العدد . وهذه الوحدات تعتبر أساسية عند كتابة قيمة أي متغير باستخدام الصيغة E ويضاف الى تلك الوحدات عدد الأرقام المطلوبة طباعتها ويتوقف ذلك على درجة التقريب المطلوب إستخراج العدد به .

### ٣-٦ إستخدام الحرف H في الجملة الشارحة :

يستخدم الحرف H (\*) في الجملة الشارحة عندما يراد كتابة أية عناوين تحتوي على حروف أو أرقام أو أية علامات خاصة . ويشرط لاستخدامها معرفة عدد الحروف والأرقام والعلامات الخاصة المراد كتابتها وكتابة ذلك العدد قبل الحرف H .

مثال ١ :

```
15 WRITE (6,15) R1, R2
1  FORMAT (23H THE REAL ROOTS ARE X1 = , F8.2, 5X,
1  7HAND X2 = , F8. 2)
```

عند تنفيذ تلك التعليمة سمنجد أن النتيجة ستكون في الصورة :

أول عمود في الطابع



```
THE REAL ROOTS ARE X1 = xxxxx.xx AND X2 = xxxxx.xx
```

\* يرمز الحرف H الى العالم الاحصائي هوليرث Hollerith إعترافاً بفضلته في إبتكار نظام البطاقات المثقوبة كوسيلة للتخزين وكان ذلك في عام ١٨٨٩ م .

نلاحظ من المثال السابق أن :

- ١ - عدد الحروف والأرقام والعلامات الخاصة المراد كتابتها أولاً هي ٢٣ ، ثم يكتب قيمة R1 وبعد ذلك يراد ترك خمس مسافات فارغة (5X) ثم يكتب عدة حروف وأرقام وعلامات خاصة أخرى عددها ٧ يتلوها مباشرة كتابة قيمة R2 .
- ٢ - إن القيم العددية التي يراد كتابتها هي قيم R1 ، R2 ولو أنها في الجملة الشارحة أخذت أسماء X1 ، X2 وهذا لن يغير من الأمر شيئاً إذ أننا في الجملة الشارحة يمكن أن نكتب ما نريده ولكن العبارة بأسماء المتغيرات الموجودة في تعليمات الكتابة ، ولذا لابد وأن تكون قيم R1 ، R2 هي المعروفة في البرنامج وليس قيم X1 ، X2 .

مثال ٢ :

```
3 | | WRITE (6,3)
    | | FORMAT (1X,10(1H*), 11HTHE#RESULTS,5(2H- +))
```

فبعد تنفيذ تلك التعليمات فستكون نتيجة الطباعة كالتالي :

أول عمود في الطابع

```
↓
*****THE#RESULTS- + - + - + - +
```

نلاحظ في هذا المثال مايلي :

- ١ - اننا استخدمنا الأقواس في الجملة الشارحة للتعبير عن التكرار ، فقبل كتابة الجملة THE RESULTS يراد كتابة العلامة \* عشر مرات ، وبعد كتابة هذه الجملة يراد كتابة العلامتين - ، + خمس مرات متتالية .
- ٢ - الجملة الشارحة السابقة تكافئ تماماً الجملة :

```
3 | | FORMAT (1X,31H*****THE#RESULTS- + - + - + - +)
```

نلاحظ من المثالين السابقين أن استخدام الحرف H في الجملة الشارحة بمائل تماماً لإستخدام الأقواس الصغيرة quote marks إلا أن الأخير أكثر تطوراً وإستخداماً حيث إستخدامها لا يتطلب معرفة عدد الحروف والأرقام والعلامات الخاصة ولكن نكتب ببساطة ما يراد كتابته بين تلك الأقواس الصغيرة . ولكن نتضح أهمية استخدام الحرف H عندما يراد كتابة مجموعة من الحروف أو الأرقام أو العلامات الخاصة عدداً معيناً من المرات . وعلى سبيل المثال اذا أردنا من الحاسب أن يطبع خطأً مكوناً من العلامة (-) عددها سبعون ، يمكننا تنفيذ ذلك ببساطة بكتابة :

15     |     WRITE (6,15)  
          |     FORMAT (1X,70 (1H -))

15     |     FORMAT (1X,35 (2H -- --))

أو

15     |     FORMAT (1X,14 (5H - - - - -))

أو

وهكذا .

٣-٧ استخدام الحرف X في الجملة الشارحة :

يستخدم الحرف X في الجملة الشارحة عندما يراد من الحاسب ترك مسافات عند القراءة أو الكتابة . فعلى سبيل المثال ، لنفترض الجزء التالي من برنامج ما :

PI = 3.14  
 R = 5. 268  
 AREA = PI\*R\*R  
 WRITE (6,18) R,PI,AREA  
 18     |     FORMAT (1X,'R=' ,F5.3,3X,'PI=' , F4.2,5X, 'AREA=' ,F7.4)

فعند تنفيذ الحاسب لهذا الجزء من البرنامج سنجد أن تنفيذ تعليمة الكتابة ستكون كالتالي :

أول عمود في الطابع  
 ↓

BR = 5.268BBBPI=3.14BBBARE A=87.1407

مثال ٢ :

A = -2.157  
 B = 13.24  
 S = A + B  
 D = B / A  
 WRITE (6,14) A,B,S  
 WRITE (6,18)A,B,D  
 14     |     FORMAT (1X,3(F6.3,2X))  
 18     |     FORMAT (1X,3(2X,F6.3))

عند تنفيذ الحاسب لهذا الجزء من البرنامج ، سنجد أن النتائج ستكون كالآتي :

أول عمود في الطابع

↓

B-2.1578813.2408811.08388

888-2.1578813.2408888-6.138

يلاحظ في هذا المثال أن هناك جملتين شارحتين أحدهما لوصف طريقة كتابة قيم المتغيرات S'B'A والأخرى للمتغيرات D'B'A . وفي كلتا الجملتين نلاحظ مايلي :

١ - بدء عملية الطباعة بترك مسافة واحدة فارغة (IX) قبل كتابة قيم المتغيرات ، وينصح بذلك عند كتابة أي بيانات أو نتائج باستخدام الطابع .

٢ - في الجملة الشارحة التي رقمها 14 نطلب فيها من الحاسب بأن يكتب قيم المتغيرات S'B'A بحيث أن قيمة كل متغير تكتب في الصورة F6.3 ثم تجميعها مسافتين فارغتين ، بينما في الجملة الشارحة التي رقمها 18 فإننا نطلب فيها من الحاسب بأن يكتب قيم المتغيرات D'B'A بحيث أن قيمة كل متغير تكتب في الصورة F6.3 وتكون مسبوبة بمسافتين فارغتين .

٣ - أننا إستخدمنا الأقواس للتعبير عن أن طريقة الطباعة الموجودة بين القوسين ستكرر عدداً من المرات يساوي 3 .

مثال ٣ :

```
5 | READ (3,5) A,B,C,D
   | FORMAT (F6.2,2X, F4.1,2(3X,F8.3))
```

فلتنفيذ تعليمة القراءة تلك ، يجب إعطاء قيم المتغيرات D'C'B'A للحاسب كما يلي :

xxx.xx    xxx.xx    xxxxxx    xxxxxx

A            B            C            D

قيمة A    قيمة B    قيمة C    قيمة D

٣-٨ استخدام علامة القسمة / (Slash) في الجملة الشارحة :

وجود العلامة / في الجملة الشارحة يعني ببساطة الانتقال الى السطر التالي سواء لقراءة بيان عند تواجدها في الجملة الشارحة الخاصة بتعليمة قراءة أو لكتابة بيان عند تواجدها في الجملة الشارحة الخاصة بتعليمة كتابة .

مثال ١ :

```

16 | READ (5,16) A,B,C,D
    | FORMAT (F8.2,2X,F6.3/F5.1, 3X, F9.3)
    | WRITE (6,18) A,B,C,D
18 | FORMAT (1X,'A=', F8.2, 'B=', F6.3, 3X, 'C=',F5.1/34X,
    | 1 'D=',F9.3/44 (1H=))

```

عند تنفيذ تعليمة القراءة لهذا الجزء من البرنامج ، فأن ذلك يتطلب اعطاء قيم المتغيرات D'C'B'A الى الحاسب لقراءتها بحيث تكون قيمتي B'A على سطر واحد وفي الصورة :

xxxx.xx**bb**xx.xxx  
 A      B      قيمة

وقيمتي D,C في السطر الذي يليه ، وفي الصورة :

xxx.x**bb**bbxxxx.xxx  
 C      D      قيمة

كما أنه عند تنفيذ تعليمة الكتابة ، فان الحاسب سيطبع قيم المتغيرات كالتالي :

أول عمود  
 ↓

**b**A = xxxxx.xx**bb**bb**b**B = xx.x**bb**bb**b**C = xxx.x D = xxxxx.xxx

مثال ٢ :

```

2 | WRITE (6,2) A, B
  | FORMAT (1X,F7.2/////1X,F9.3)

```

في هذا المثال يراد من الحاسب كتابة قيمة المتغير A في سطر في الصورة xxxx.xx وبعد ذلك يترك أربعة أسطر ثم يكتب قيمة المتغير B في الصورة xxxx.xxx .

وفي المثالين السابقين نلاحظ عدم وجود فاصلة (و) قبل أو بعد العلامة / .



## الفصل الرابع

# تعليمات التحكم



## الفصل الرابع

«Control Statements»

تعليمات التحكم :

مقدمة :

علما فيما سبق أن البرنامج المكتوب بلغة ما هو عبارة عن مجموعة من التعليمات التتابعية والمتتالية والتي تهدف في النهاية الى إيجاد القيمة العددية لعملية حسابية قد تكون بسيطة أو معقدة ، وقد يتم حسابها في تعليمة واحدة أو بعد تجزئتها الى مجموعة تعليمات مستقلة ثم يتم تجميعها لتعطي في النهاية قيمة العملية الحسابية . فعلى سبيل المثال :

مثال ١ :

إذا أردنا حساب قيمة المتغير Y والذي يتم حسابه عن طريق المعادلة التالية :

$$Y = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

فإن ذلك يمكن أن يتم مثلاً عن طريق :

١ - كتابة المعادلة السابقة بلغة الفورتران :

$$Y = (-B + (B*B - 4.*A*C)**0.5) / (2.*A)$$

٢ - تجزئة مكونات المعادلة واعادة تجميعها بعد حساب تلك المكونات كلا على حدة :

$$\begin{cases} R = B*B - 4.*A*C \\ A2 = 2.*A \\ S = -B + R**0.5 \\ Y = S/A2 \end{cases}$$

والطريقتان السابقتين ستعطيان نتيجة واحدة لقيمة المتغير Y ، اذا كانت قيم المتغيرات C,B,A مناسبة . أي اذا كانت قيمة A لاتساوي الصفر وكانت قيمة (B\*B-4.\*A\*C) غير سالبة . فاذا

كانت قيمة A2 تساوي صفراً فسنجد أن قيمة Y نظرياً في الطريقة الأولى تساوي مالا نهاية ، كما أن قيمة A2 في الطريقة الثانية تساوي صفراً وقيمة Y ( نظرياً ) متساوي مالا نهاية . وكما نعلم فإن القيمة ( مالا نهاية ) هي قيمة غير معلومة يختلف تقديرها من شخص إلى آخر ، ولكن هل يستطيع الحاسب إعطاؤنا نتيجة محددة لقيمة المتغير Y في الطريقتين السابقتين ؟ والاجابة على ذلك سيكون بالنفي ، إذ أن الحاسب أيضاً لن يتمكن من تقدير القيمة مهما كان الحاسب كبيراً .

كذلك سنصل إلى نفس الاحابة من الحاسب اذا كانت قيمة  $(B*B - 4.*A*C)$  سالبة ، حيث أن الحاسب لن يعطينا نتيجة محددة لجذر كمية سالبة . ومثل هذه الحالات يقوم الحاسب باكتشافها في مرحلة التنفيذ للبرنامج «Execution» والتي كما سبق أن عرفنا أنها المرحلة التي تلي ترجمة البرنامج واكتشاف الأخطاء اللغوية به (Compilation) . ولذا فإن معظم الحاسبات تعتبر ظهور تلك الحالات كنوع آخر من الأخطاء Errors يقوم الحاسب عند اكتشافها في البرنامج بالتوقف عن تنفيذ التعليمات التي تليها وتحذير واضع البرنامج من نوع الخطأ الذي وقع فيه .

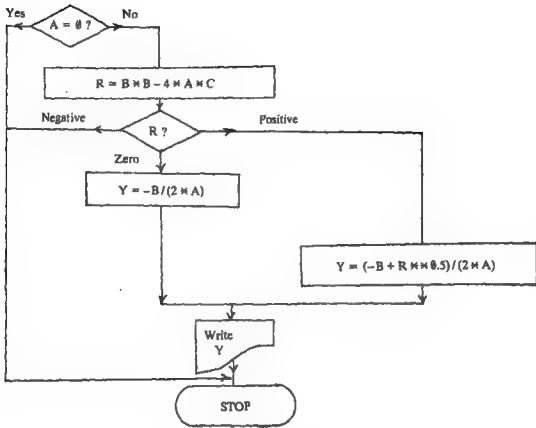
ولعدم الوقوع في مثل تلك الأخطاء عند كتابة برنامج وتنفيذه وجب على المبرمج أن يعطي الحاسب التعليمات المناسبة التي تجعله يتحاشى الوقوع في مثل تلك الأخطاء وذلك بأن يعطي الحاسب التعليمات المناسبة التي تجعله :

١ - يقوم باختيار المتغير أو الكمية التي يمكن أن تكون سبباً في الوقوع في أحد أنواع تلك الأخطاء .

٢ - يتوجه إلى التعليمة أو مجموعة التعليمات المناسبة في البرنامج ، دون أن نجعل الحاسب يقوم بتنفيذ التعليمة التي تكون سبباً في ظهور مثل هذا النوع من الأخطاء .

ففي المثال السابق ، اذا كان المبرمج خذراً ومهراً كان عليه أن يتساءل عن قيمة A وهل هي صفرية أم لا ؟ وكذلك عن قيمة  $(B*B - 4.*A*C)$  وهل هي سالبة أم لا ؟ وماذا عليه أن يفعل في كل حالة ؟

ولنبداً أولاً بعمل مخطط تدفق لتلك المشكلة نحدد فيه ما يجب علينا عمله وماهي العمليات الحسابية التي يتطلب القيام بها ؟ نفرض أن قيم  $C, B, A$  معروفة لدينا ( سواء سبق حسابها أو قراءتها في البرنامج ) ، فإن أول ما علينا عمله هو اختبار قيمة المتغير A ومعرفة هل هي صفرية أم لا ؟ فإذا كانت  $A = 0$  ، وجب علينا إعطاء الحاسب أمراً بعدم تنفيذ بقية البرنامج والذهاب إلى تعليمة التوقف .



شكل (١-٤)

- وإذا كانت قيمة المتغير  $A$  غير صفرية ، وجب علينا أن نعرف قيمة المقدار  $R = B^2 - 4AC$  وهل هي سالبة أم صفرية أم موجبة .

- فإذا كانت  $R$  سالبة ( $R < 0$ ) ، وجب علينا أيضا الذهاب الى تعليمة التوقف عن تنفيذ بقية البرنامج لأننا في هذه الحالة لن نتمكن من حساب قيمة الجذر التربيعي لكمية سالبة ، كما سبق أن أشرنا .

- وإذا كانت قيمة  $R$  صفرية ( $R = 0$ ) ، فلن يكون هناك داع لجعل الحاسب يأخذ وقتاً في حساب الجذر التربيعي لكمية صفرية ، وخاصة اذا علمنا أن كل دقيقة على الحاسب تكلفنا الكثير مادياً . وبالتالي فيمكننا الاكتفاء بحساب قيمة  $Y$  بأن نجعل الحاسب يذهب الى التعليمة :

$$Y = -B / (2.*A)$$

- فإذا كانت قيمة  $R$  موجبة ( $R > 0$ ) فسنقوم بحساب قيمة  $Y$  بجعل الحاسب ينتقل الى التعليمة :

$$Y = (-B + R**0.5) / (2.*A)$$

## مثال ٢ :

- لنفرض أن المتغير  $Y$  يتم حسابه على النحو التالي :

i)  $Y = (x-a)^3 - (x-a)^2 + (x-a) + 5$ .

إذا كانت قيمة  $x$  أكبر من قيمة  $a$  . عندما تكون  $(x > a)$  .

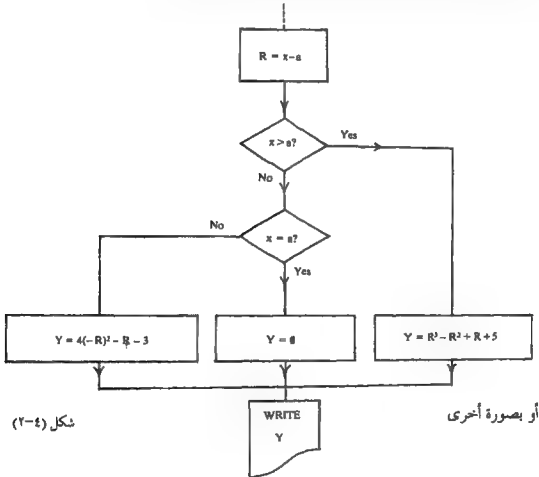
ii)  $Y = 4(a-x)^2 + (a-x) - 3$

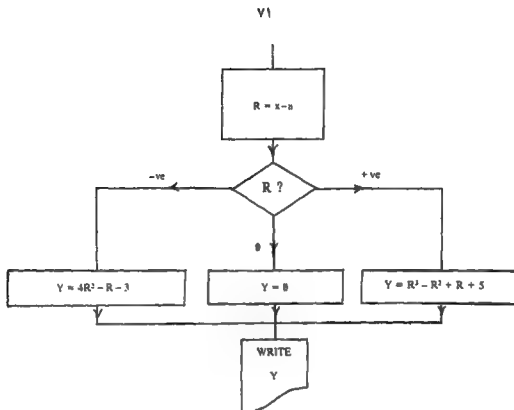
إذا كانت قيمة  $x$  أصغر من قيمة  $a$  . أي عندما تكون  $(x < a)$

iii)  $Y = 0$

إذا كانت قيمة  $x$  تساوي قيمة  $a$  . أي عندما تكون  $(x = a)$

، فإذا افترضنا مرة أخرى أن قيم  $a, x$  معروفة لدينا ( سواء سبق حسابها أو قراءتها في البرنامج ) ، فإن قيمة  $Y$  سيتوقف حسابها على قيم  $a, x$  وهل هما متساويتان أم أن أحدهما أكبر من الأخرى . ولذا فإن مخطط التدفق شكل (٢-٤) لحل تلك المشكلة قد يكون كالآتي :





شكل (٣-٤)

من المثالين السابقين نلاحظ أننا قد نضطر في بعض الأحيان إن لم تكن أغلبها ، أن نقطع انسياق تسلسل تنفيذ التعليمات المتتالية في البرنامج ونتيجة الى تعليمة أخرى في موضع آخر في البرنامج قد تسبق أو تلي آخر تعليمة قام الحاسب بتنفيذها لتكون نقطة بدء مجموعة من التعليمات المتعاقبة يبدأ الحاسب في تنفيذها الواحدة تلو الأخرى الى أن يواجه مرة أخرى بتعليمة شرطية اذا ، تضطر الحاسب الى الانتقال الى موضع آخر في البرنامج وهكذا .

وفي لغة الفورتران فان ذلك يتم باستخدام تعليمات معينة تسمى تعليمات التحكم والانتقال  
Control and Branch Statements

#### ١-٤ تعليمة التحكم ..... اذا IF

هناك صورتين عامتين مختلفتين لهذه التعليمة في لغة الفورتران ، وان كانتا متشابهتين في عملهما . والصورة العامة الأولى لتعليمة التحكم ..... اذا IF هي :

#### ١-٤-١ تعليمة اذا الحسائية :

nnnnn	IF (Arithmetic Expression) n1, n2, n3
-------	---------------------------------------

ويطلق على هذه التعليمة اسم تعليمة اذا الحسابية

حيث :

- nnnnn : رقم تعليمة اذا في البرنامج ( ان وجدت )

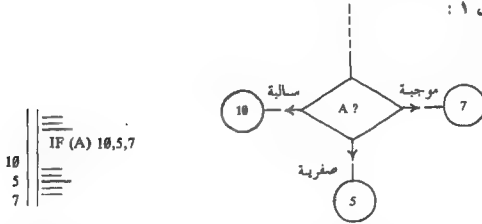
- Arithmetic Expression : ترمز الى تعبير رياضي مطلوب حساب قيمته ومعرفة ما اذا كان موجباً أم صفرىاً أم سالباً . وهذا التعبير قد يكون متغيراً بسيطاً حقيقياً أو صحيحاً أو تعبيراً رياضياً يحتوي على أكثر من متغير - بشرط أن تكون جميعها حقيقة أو صحيحة . وليس خليطاً منها Mixed mode .

- n1 : رقم التعليمة التي تطلب فيها من الحاسب أن يذهب اليها في حالة ما اذا كانت قيمة التعبير الرياضي سالبة .

- n2 : رقم التعليمة التي تطلب فيها من الحاسب أن يذهب اليها في حالة ما اذا كانت قيمة التعبير الرياضي صفرياً ، أي تساوي صفراً .

- n3 : رقم التعليمة التي تطلب فيها من الحاسب أن يذهب اليها في حالة ما اذا كانت قيمة التعبير الرياضي موجبة .

مثال ١ :



أي أنه يراد من الحاسب أن يذهب الى التعليمة التي رقمها 10 ، اذا كانت قيمة المتغير A سالبة ، وأن يذهب الى التعليمة التي رقمها 5 ، اذا كانت قيمة A تساوي صفراً ، وأن يذهب الى التعليمة التي رقمها 7 اذا كانت قيمة المتغير A موجبة .

نلاحظ من المثال السابق مايلي :

- ١ - التعبير الرياضي الموجود بين القوسين عبارة عن متغير بسيط قد يكون حقيقياً أو صحيحاً .
- ٢ - أرقام التعليمات لايلزم أن تكون مرتبة تصاعدياً أو تنازلياً .



مثال ٢ :

$$\begin{array}{c|c} & \text{IF } (B*B-4.*A*C) \text{ 6,8,5} \\ \hline 8 & \\ 5 & \\ 6 & \end{array}$$

أنه إذا كانت قيمة التعبير الرياضي الموجود ما بين القوسين :

- . سالبة : فيجب الانتقال الى التعليمة التي رقمها 6 ،
- . صفراً : فيجب الانتقال الى التعليمة التي رقمها 8 ،
- . موجبة : فيجب الانتقال الى التعليمة التي رقمها 5 ،

نلاحظ في هذا المثال مايلي :

- ١ - التعبير الرياضي الموجود بين القوسين عبارة عن عملية حسابية قد تحتوي على عمليات جمع وطرح وضرب وقسمة ... الخ .
- ٢ - عدم ظهور أرقام التعليمات في بقية البرنامج بنفس ترتيب ظهورها في تعليمة اذا IF .
- ٣ - إن الحاسب سيقدر الانتقال الى التعليمة 6 أو التعليمة 8 أو التعليمة 5 ، بعد حساب التعبير الرياضي  $(B*B-4.*A*C)$  ومعرفة قيمته .

مثال ٣ :

$$\begin{array}{c|c} & \text{IF } (A) \text{ 5,3,5} \\ & Y = -B/(2.*A) \\ \hline 5 & \\ & \end{array}$$

يتبين من هذا المثال أنه اذا كانت قيمة المتغير A سالبة أو موجبة ( أي غير صفرية ) ، فإن الحاسب سينتقل الى التعليمة التي رقمها 5 . أما اذا كانت قيمة المتغير A صفرية فسينتقل الحاسب الى التعليمة التي رقمها 3 . ولذلك فمن الممكن أن يتساوى رقمين في تعليمة التحكم اذا ، بينما يكون الرقم الثالث مختلفاً . وعلى سبيل المثال مايلي :

$$\left| \begin{array}{c} \text{IF } (S - 3.) \text{ 4,7,4} \end{array} \right|$$

أي إذا كانت قيمة S تساوي 3 ، فعل الحاسب أن يتوجه الى التعليمة التي رقمها 7 . أو بمعنى آخر ، إذا كانت قيمة S لائساوي 3 ، فعل الحاسب أن ينتقل الى التعليمة التي رقمها 4 .

مثال :

$$\left| \begin{array}{c} \text{IF } (S - 3.) \text{ 6,6,8} \end{array} \right|$$

أي أنه إذا كانت قيمة S اكبر من 3 ، ( أي أن  $(S - 3)$  كمية موجبة ) فعل الحاسب أن ينتقل الى التعليمة التي رقمها 8 والا فيجب الانتقال الى التعليمة التي رقمها 6 .

$$\left| \begin{array}{c} \text{IF } (S - 3.) \text{ 4,2,2} \end{array} \right|$$

أي أنه إذا كانت قيمة S اكبر من أو تساوي 3 ، فعل الحاسب أن ينتقل الى التعليمة التي رقمها 2 . أو بمعنى آخر ، إذا كانت قيمة S أقل من 3 فإن الحاسب سينتقل الى التعليمة التي رقمها 4 .

٤-١-٢ تعليمة إذا المنطقية :

والصورة العامة الثانية لتعليمة التحكم .... إذا IF هي :

nnnnn	IF (Logical Expression) es
-------	----------------------------

حيث :

- nnnnn : رقم تعليمة إذا ... IF في البرنامج ( ان وجدت ) .
- Logical Expression : تعبير منطقي ينقسم في الحقيقة الى جزئين يراد المقارنة بينهما ، من حيث التساوي أو عدمه أو أن احدهما اكبر من الآخر أو أصغر منه . فإذا افترضنا أن جزئي التعبير المنطقي هما Exp1 , Exp2 فيمكن كتابة تعليمة إذا .. IF في إحدى الصور الست

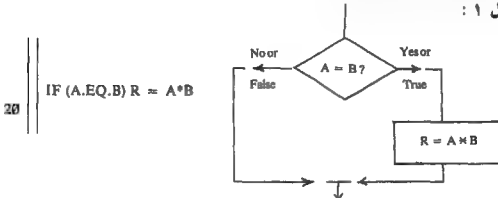
التالية :

- ١ -  $\text{IF (Exp1.EQ.Exp2) es}$   
أي إذا كان (Exp 1. Equals يساوي Exp 2)
- ٢ -  $\text{IF (Exp 1. GE. Exp 2) es}$   
أي إذا كان (Exp 1. Greater or Equal أكبر من أو يساوي Exp 2)
- ٣ -  $\text{IF (Exp 1 .GT. Exp 2) es}$   
أي إذا كان (Exp 2 . أكبر من Exp 1 Greater Than )
- ٤ -  $\text{IF (Exp 1. LE. Exp 2)}$   
أي إذا كان (Exp. 2 . أقل من أو يساوي Exp 1. Less or Equal )
- ٥ -  $\text{IF (Exp 1. LT. Exp 2) es}$   
أي إذا كان (Exp 2 . أقل من Exp 1. Less Than )
- ٦ -  $\text{IF (Exp 1 .NE. Exp 2) es}$   
أي إذا كان (Exp. 2 . لا يساوي Exp 1. Not Equal )

وقد يكون كل من التعبيرين Exp 2 , Exp 1 عبارة عن متغيرات بسيطة ، حقيقة أو صحيحة أو تعبيرات رياضية تحتوي على أكثر من متغير بشرط أن تكون جميعها من نوع واحد ، أما حقيقة أو صحيحة وليس خليطاً منها .

- es : تعليمة تنفيذية Executable Statement يقوم الحاسب بتنفيذها إذا تحقق الشرط الذي يربط بين التعبيرين Exp 2 , Exp 1 وإذا لم يتحقق ذلك الشرط فإن الحاسب يقوم بتنفيذ التعليمة التي تلي تعليمة التحكم إذا .... IF مباشرة في البرنامج . ويطلق على هذه التعليمة اسم تعليمة إذا المنطقية Logical IF Statement .

مثال ٩ :



أي أنه إذا كانت قيمتي  $B, A$  متساويتين ، فإن الحاسب سيقوم بحساب قيمة  $R$  ثم يبدأ في تنفيذ التعليمات التي تلي تعليمة إذا ... IF مباشرة أما إذا كانت قيمتي  $B, A$  مختلفتين ، فإن الحاسب لن يقوم بحساب قيمة  $R$  بل سيتجه مباشرة الى التعليمة التالية والتي رقمها 20 .

مثال ٢ :

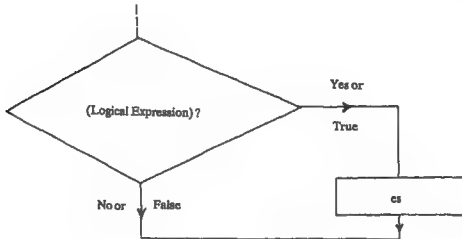
```

IF ((B*B - 4.*A*C). GE.0.) Y = (- B + (B*B - 4.*A*C)**0.5)/(2.*A)
STOP

```

يتبين من هذا المثال أنه إذا كانت قيمة  $(B^2 - 4AC)$  اكبر من أو تساوي الصفر ، فإن الحاسب سيقوم بتنفيذ بقية التعليمة ، أي حساب قيمة  $Y$  ثم يتوقف . أما إذا كانت قيمة  $(B^2 - 4AC)$  سالبة ، أي أقل من الصفر فإن الحاسب سيتوقف دون حساب قيمة  $Y$  .

ومن المثالين السابقين يتبين أن تنفيذ بقية تعليمة إذا ... IF يتوقف على تحقق العلاقة المنطقية التي تربط بين جزئي التعبير المنطقي الموجود بين القوسين . كما يتبين أيضا أن الحاسب سيقوم بتنفيذ التعليمة التي تتلو تعليمة إذا ... IF المنطقية مباشرة ، بصرف النظر عن تنفيذ بقية التعليمة أم لا فيما عدا إذا كانت بقية تعليمة إذا هي إحدى تعليمات الانتقال Branch Statements التي سنشرحها فيما بعد . وبذلك يمكن أن نمثل رسم مخطط تدفق البيانات شكل (٤-٤) الخاص بتلك التعليمة كالآتي :



شكل (٤-٤)

#### ٤-١-٣ ملحوظات على تعليمتى التحكم اذا .... IF

(أ) في تعليمة التحكم اذا ... IF الحسابية :

- ١ - لا بد من كتابة التعبير الحسابي بين قوسين .
- ٢ - أن يكون التعبير الحسابي متجانس في تكوينه وليس خليطاً من متغيرات صحيحة وحقيقية أو ثوابت متغيرة floating وثابته fixed .
- ٣ - في تعليمة اذا الحسابية يجب الالتزام بتسلسل أرقام التعليمات ، بحيث أن رقم التعليمة الأولى n1 مستخلص كى ينتقل اليها الحاسب في حالة ما اذا كان التعبير الرياضي سالباً ، ورقم التعليمة الثانية n2 اذا كانت قيمة التعبير الرياضي صفراً . ورقم التعليمة n3 في حالة ام اذا كان التعبير الرياضي موجبا .
- ٤ - التحقق من وجود فاصلة Comma بين كل رقمى تعليمة .
- ٥ - أن أرقام التعليمات الثلاث قد تكون مختلفة أو تتساوى التين منها .
- ٦ - أن أرقام التعليمات يجب أن تكون أعداداً صحيحة غير كسرية ولا تحتوي على اشارات سالبة أو موجبة .
- ٧ - الا تكون أرقام التعليمات متغيرات سواء حقيقية أو صحيحة ، حتى ولو سبق اعطاء قيم لتلك المتغيرات . فعلى سبيل المثال ، لا يمكن القول :

$\begin{array}{l} K = 5 \\ IF (c1.EQ.c2) GOTO K \end{array}$	$\begin{array}{c} 5 \\ 7 \\ 10 \end{array}$	$\begin{array}{l} K = 5 \\ L = 7 \\ IF ( \quad ) L, 10, K \end{array}$
--	---	--

(ب) في تعليمة التحكم اذا ... IF المنطقية :

- ١ - يجب أن يكون جزئى التعبير المنطقي Exp 1, Exp 2 المراد مقارنتهما من نوع واحد ، بحيث لا يجب أن يكون احدهما محتويًا على متغيرات حقيقية والآخر على متغيرات صحيحة . وبالتالي أيضاً لا يجب أن يحتوي أحدهما على خليط من المتغيرات الصحيحة والحقيقية .
- ٢ - التحقق من وجود رمز المقارنة بين جزئى التعبير المنطقي بين نقطتين .

٣ - سيقوم الحاسب بتنفيذ التعليمة التي تتلو تعليمة اذا المنطقية مباشرة ، مهما كانت نتيجة المقارنة بين جزئى التعبير المنطقي (Exp 2, Exp 1) الا اذا كانت التعليمة التنفيذية es المكملة لتعليمة إذا المنطقية هي تعليمة انتقال GOTO والتي سيأتي الحديث عنها في الفصل التالي .

## الفصل الخامس

### تعليمات الإنتقال





## الفصل الخامس

### Transfer Statements

### تعليمات الانتقال

#### مقدمة : تعليمة الانتقال ... GOTO

من تعليمات التحكم التي سبق شرحها ، تبين لنا أننا قد نضطر في بعض الأحيان الى قطع تسلسل تنفيذ مجموعة من التعليمات والاتجاه بالحاسب الى تنفيذ مجموعة أخرى من التعليمات في موضع آخر من البرنامج قد يسبق موضع هذا القطع أو قد يأتي بعده . وفي جميع الأحوال كانت عملية قطع تسلسل تنفيذ تلك التعليمات والانتقال بالحاسب الى تنفيذ تعليمات أخرى في موضع آخر من البرنامج ، تتم وفقا للتعليمة اذا IF التي تقوم باختبار إشارة كمية ما ، والتوجه بالحاسب طبقا لإشارة تلك الكمية .

وفي لغة الفورتران ، يمكن الانتقال أيضا بالحاسب من تنفيذ مجموعة تعليمات في البرنامج الى مجموعة أخرى في مكان آخر من نفس البرنامج باستخدام تعليمات انتقال Branching Statements تنقسم الى نوعين :

#### ١-٥ تعليمة الانتقال GOTO الغير مشروطة Un-conditional

والصورة العامة لتلك التعليمة هي :

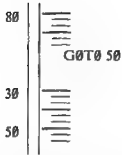
vvvvv	GOTO n
-------	--------

حيث :

- n هي رقم التعليمة في البرنامج ، والتي يراد من الحاسب الذهاب اليها لتنفيذها وتنفيذ التعليمات التي تليها ، ولا بد أن تكون n رقما صحيحا موجبا وليس اسما لمتغير ، وقد تكون التعليمة التي رقمها n سابقة لتعليمة الانتقال أو لاحقة لها .

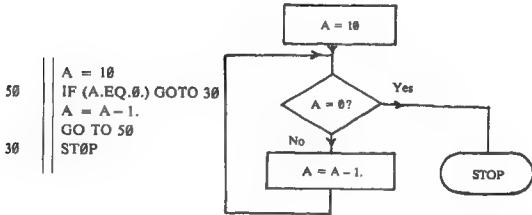
- vvvvv وهي رقم تعليمة الانتقال GOTO الغير مشروطة في البرنامج . وفي الحقيقة فإن هذا الرقم ليس له داع في هذه التعليمة بالذات ( أنظر ملحوظات على تعليمتى الانتقال ) .

## مثال ٥-١-١ :



بعد أن يقوم الحاسب بتنفيذ التعليمة التي رقمها 80 ، يبدأ في تنفيذ التعليمات التي تليها مباشرة حتى يصل الى تعليمة الانتقال الغير مشروطة GOTO 50 والتي تسبب في أن ينتقل الحاسب من هذا الموضع من البرنامج الى الموضع الذي يبدأ بالتعليمة التي رقمها 50 دون تنفيذ التعليمة التي رقمها 30 والتعليمات التي تسبقها أو تليها .

## مثال ٥-١-٢ :



في المثال السابق نجد أن هناك تعليمتي انتقال احدهما الى التعليمة رقم 30 ، والأخرى الى التعليمة رقم 50 .

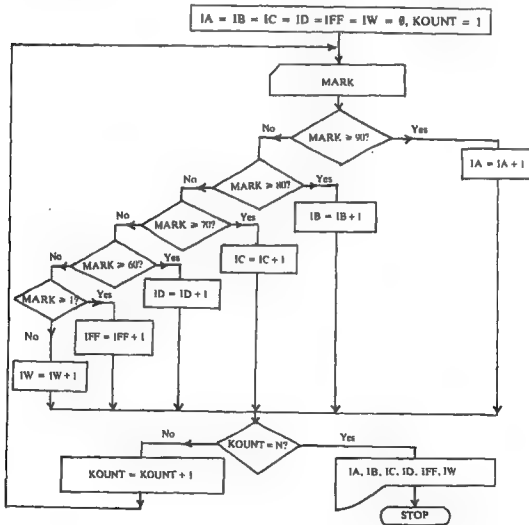
والتعليمة رقم 50 في هذا المثال تشمل تعليمتي التحكم والانتقال في آن واحد ، ولو أن تنفيذ الشق الثاني من هذه التعليمة (GOTO 30) سيتحقق عندما تصل قيمة A الى الصفر والتي عندها سينتقل الحاسب الى التعليمة التي رقمها 30 ، حيث يتوقف STOP عن تنفيذ بقية التعليمات التي تلي تلك التعليمة .

## مسال ٥-١-٣ :

يفرض أن هناك عدد  $N$  من الطلبة أدوا امتحانا في مادة ما ، وحسب درجة MARK كل منهم

يراد معرفة :

- ١ - عدد الطلبة الذين حصلوا على درجة ممتاز  $(90 \leq \text{MARK} \leq 100)$
- ٢ - عدد الطلبة الذين حصلوا على درجة جيد جداً  $(80 \leq \text{MARK} \leq 89)$
- ٣ - عدد الطلبة الذين حصلوا على درجة جيد  $(70 \leq \text{MARK} \leq 79)$
- ٤ - عدد الطلبة الذين حصلوا على درجة مقبول  $(60 \leq \text{MARK} \leq 69)$
- ٥ - عدد الطلبة الراسبون  $(1 \leq \text{MARK} \leq 59)$
- ٦ - عدد الطلبة المنسحبون  $(\text{MARK} = 0)$



شكل (٥-١) مخطط التدفق للمسألة أعلاه

- بفرض أن عدد الطلبة من الفئة الأولى يمثلها المتغير IA ،  
 عدد الطلبة من الفئة الثانية يمثلها المتغير IB ،  
 عدد الطلبة من الفئة الثالثة يمثلها المتغير IC ،  
 عدد الطلبة من الفئة الرابعة يمثلها المتغير ID ،  
 عدد الطلبة من الفئة الخامسة يمثلها المتغير (٥) IFF ،  
 عدد الطلبة من الفئة السادسة يمثلها المتغير IW ،

```

C      PUTING THE GRADE POSITIONS TO ZERO.
      IA = 0
      IB = 0
      .
      .
      IW = 0
C      STARTING COUNT THE NUMBER OF STUDENTS.
      KOUNT = 1
5      READ (7, 10) MARK
10     FORMAT (I3)
C      TESTING THE MARK AND PUTING IT IN THE APPROPRIATE
C      PLACE.
      IF (MARK. GE. 90) GOTO 35
      IF (MARK. GE. 80) GOTO 30
      IF (MARK. GE. 70) GOTO 25
      IF (MARK. GE. 60) GOTO 20
      IF (MARK. GE. 1) GOTO 15
      IW = IW + 1
      GOTO 100
  
```

(٥) عند اختيار اسماء المتغيرات يجب أن تكون تلك الأسماء مختلفة عن تعليمات الفورتران المعروفة مثل IF أو GOTO أو STOP أو ... الخ . كذلك يلاحظ أن جميع أسماء المتغيرات تبدأ بحرف I وذلك كي تكون قيم تلك المتغيرات صحيحة أي لا تقبل فيما كسرية . وهو ما ينفق منطقيا مع ما تمثله تلك المتغيرات حيث أن كلا منها يمثل عددا للطلبة لا يحتوي على قيم كسرية .

في مخطط التدفق السابق نلاحظ أننا بعد عمل الاختبارات المختلفة على درجة الطالب ووضعها في المكان المناسب لها ، نعاود قراءة الدرجة التي تليها . ويتحكم في تلك العملية متغيراً يحمل كمعاد KOUNT يتزايد بعد قراءة كل درجة بمقدار الوحدة الى أن تصل قيمته الى N ، ويمكن تصور البرنامج الذي يعبر عن مخطط التدفق كما يلي :

```

15  IFF = IFF + 1
    GOTO 100
20  ID = ID + 1
    GOTO 100
25  IC = IC + 1
    GOTO 100
30  IB = IB + 1
    GOTO 100
35  IA = IA + 1
100 IF (KOUNT. EQ. N) GOTO 150
    KOUNT = KOUNT + 1
    GOTO 5
C   WRITING THE RESULTS.
150 WRITE (7, 200) IA,IB,IC,ID,IFF,IW
200 FORMAT (6I8)
    STOP
    END

```

#### ٥-٢ تعليمية الانتقال GOTO المشروطة أو ( المحسوبة ) : Computed GOTO

علمنا مما سبق أن تعليمة اذا IF الحسابية يمكن أن تنتقل بالحاسب الى ثلاث مواضع مختلفة على الأكثر من البرنامج ( الانتقال الى موضع ما في البرنامج اذا كان المتغير أو التعبير سالبا والى موضع ثان عندما يكون المتغير أو التعبير قيمته صفرا والى ثالث عندما يكون موجبا ) . كذلك فإن تعليمة الانتقال الغير مشروطة يمكن أن تنتقل بالحاسب الى موضع واحد فقط في البرنامج يبدأ بتعليمة لها رقم يتفق مع الرقم المصحوب بتعليمة الانتقال GOTO الغير مشروطة .

وهناك تعليمة انتقال أخرى يمكن باستخدامها التنقل الى اكثر من موضع في البرنامج ، ويطلق على هذه التعليمة اسم تعليمة الانتقال GOTO المشروطة . والصورة العامة لتلك التعليمة هي :

vvvvv	GOTO (n <sub>1</sub> , n <sub>2</sub> , n <sub>3</sub> ,..., n <sub>m</sub> ), iv
-------	---

حيث :

vvvvv - رقم تعليمة الانتقال GOTO المحسوبة في البرنامج ، وهذا الرقم إختيارى يتوقف وجوده أو عدم وجوده على واضع البرنامج .

-  $n_1, n_2, n_3, \dots, n_m$  : أرقام تعليمات لابد من تواجدها في البرنامج حيث قد يتطلب الأمر الانتقال إليها خلال تنفيذ الحاسب للبرنامج ، وبالطبع لابد أن تكون تلك الأرقام صحيحة ولا يحتوي على أية كسور . وقد تكون جميع هذه الأرقام مختلفة أو أن بعضها متشابه ، ولابد من تواجد هذه الأرقام بين قوسين وكذلك لابد أيضا من تواجد فاصلة Comma بين كل رقم وآخر .

- iv : متغير صحيح لابد أن يكون قد سبق إعطاؤه إحدى القيم 1 أو 2 أو ... أو m . حيث أنه بناء على قيمة ذلك المتغير سيتنقل الحاسب إلى التعليمة رقم  $n_1$  أو  $n_2$  أو ... أو  $n_m$  . كذلك يجب ملاحظة وجود فاصلة Comma قبل كتابة المتغير الصحيح .

وعندما يقوم الحاسب بتنفيذ تلك التعليمة ، فإنه ينتقل إلى الجزء من البرنامج الذي يبدأ بالتعليمة التي رقمها  $n_1$  إذا كانت قيمة التعبير iv هي 1 ، وإلى الجزء من البرنامج الذي يبدأ بالتعليمة التي رقمها  $n_2$  إذا كانت قيمة المتغير iv هي 2 وهكذا . وبالتالي فإننا نتوقع أن قيمة المتغير iv في أي موضع من البرنامج لن يزيد عن m .

مثال ٥-٢-١ :

- قيمة K التي ستأخذ إحدى القيم 1 أو 2 أو 3 أو 4  $K = \dots$

30 |

10 |

GOTO (15, 10, 30, 10), K → التعليمة الانتقال المحسوبة والتي  
بناء على قيمة K سيتوجه الحاسب إلى التعليمة 15 أو 10 أو 30

تبعاً لقيمة K . ففي هذا المثال إذا كانت قيمة K تساوي 1 ، فإنه عند تنفيذ الحاسب لتعليمة الانتقال المحسوبة سينتقل إلى الموضع من البرنامج الذي يبدأ بالتعليمة رقم 15 .  
وإذا كانت K تساوي 2 فإن الحاسب سينتقل إلى التعليمة رقم 10 وهكذا ، ونلاحظ في هذا المثال :

- ١ - إن الحاسب سينتقل إلى الموضع من البرنامج الذي يبدأ بالتعليمة رقم 10 عندما تأخذ K القيمة 2 أو 4 ، أي أن أرقام التعليمات بين القوسين قد تكون جميعها مختلفة أو بعضها متشابهة .
- ٢ - أن قيمة K في ذلك البرنامج لن تزيد عن 4 ، حيث لا يوجد إلا أربعة أرقام تعليمات بين القوسين حتى وأن تساوي فيهما اثنين أو أكثر .
- ٣ - قد تكون أرقام التعليمات الموجودة بين القوسين أرقاماً لتعليمات تسبق أو تأتي بعد تعليمة الانتقال المحسوبة نفسها .

## مثال ٥-٢-٢ :

باستخدام تعليمة الانتقال GOTO المحسوبة ، اكتب برنامجا لحساب .

- ١ - مجموع الأعداد الفردية الواقعة بين 1 و N  $(1+3+5+...+N)$   
 ٢ - مجموع الأعداد الزوجية الواقعة بين 1 و N  $(2+4+6+...+N)$

بافتراض أن مجموع الأعداد الفردية يمثل المتغير SODD (Sum of ODD nos.)  
 وأن مجموع الأعداد الزوجية يمثل المتغير SEVEN (Sum of EVEN nos.)  
 ولاجراء أي عمليات تجمع في أي متغير يلزم أن نضع في هذا المتغير القيمة صفرا .  
 أي نضع  $SODD = 0$  ،  $SEVEN = 0$  وتكمل البرنامج كالآتي :

	$L = 1 \rightarrow$	عداد يبدأ بالقيمة 1
	$SEVEN = 0. \rightarrow$	تصغير المتغير الذي سيم فيه جميع الأعداد الزوجية
	$SODD = 0. \rightarrow$	تصغير المتغير الذي سيم فيه جميع الأعداد الفردية
	$K = 1 \rightarrow$	متغير للتحكم في تعليمة الانتقال GOTO المحسوبة
100	GOTO (10, 20), K	
10	SODD = SODD + FLOAT (L)	
	GOTO 30	
20	SEVEN = SEVEN + FLOAT (L)	
	$K = 0$	
30	IF (L. EQ. N) GOTO 150	
	$L = L + 1$	
	$K = K + 1$	
	GOTO 100	
150	WRITE (7, 200) SODD, SEVEN	
200	FORMAT (1X, 2f14.)	
~	STOP	
	END	

## مثال ٥-٢-٣ :

في امتحان ما ، تم عمل ترميم لمجموعة عددها N من الطلبة والطالبات بحيث أن لكل طالب أو طالبة رقمين ، أحدهما للتعريف بالنوع ( ذكر أو أنثى ) بحيث يخصص الرقم 1 للذكور والرقم 2 للإناث . والآخر للتعريف بتقدير الطالب أو الطالبة بحيث يخصص :  
 الرقم 1 لكل طالب أو طالبة حاصل على درجة امتياز ،

- الرقم 2 لكل طالب أو طالبة حاصل على درجة جيد جدا ،  
الرقم 3 لكل طالب أو طالبة حاصل على درجة جيد ،  
الرقم 4 لكل طالب أو طالبة حاصل على درجة مقبول ،  
الرقم 5 لكل طالب أو طالبة راسب في الامتحان ،  
الرقم 6 لكل طالب أو طالبة منسحب من الامتحان .

والمطلوب حساب :

- ١ - عدد الطلبة الذكور .  
٢ - عدد الطلبة الاناث .  
٣ - عدد الطلبة والطالبات في كل درجة من الدرجات الست السابقة .

بافتراض أن عدد الطلبة الذكور سيمثله المتغير (Number of MALES) NMALES

وأن عدد الطلبة الاناث سيمثله المتغير (Number of FEMALES) NFEMAL

وأن عدد الطلبة والطالبات في كل درجة من الدرجات الست الباقية يمثلها المتغيرات :

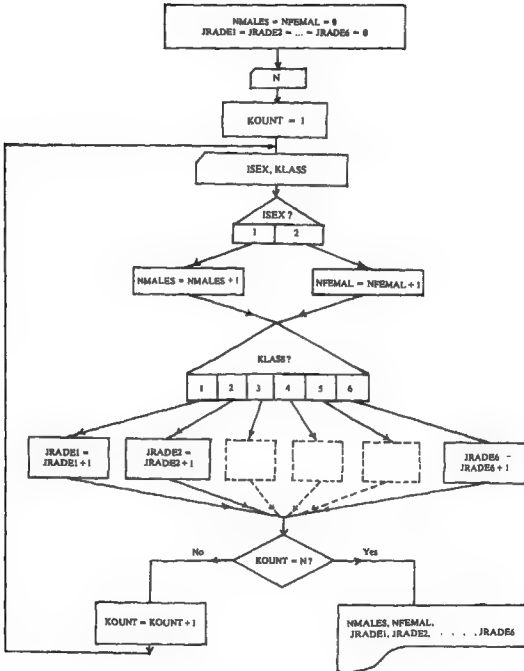
- 1 JRADE مجموع الطلبة والطالبات الذين حصلوا على تقدير امتياز ،  
2 JRADE مجموع الطلبة والطالبات الذين حصلوا على تقدير جيد جدا ،  
3 JRADE مجموع الطلبة والطالبات الذين حصلوا على تقدير جيد ،  
4 JRADE مجموع الطلبة والطالبات الذين حصلوا على تقدير مقبول ،  
5 JRADE مجموع الطلبة والطالبات الذين راسبوا في الامتحان ،  
6 JRADE مجموع الطلبة والطالبات الذين انسحبوا من الامتحان .

وإذا فكرنا فم يجب علينا اعطاؤه للحاسب كي يتمكن من حساب ماهو مطلوب منا ، نجد أن هناك قيمتين لابد أن يعرفهما الحاسب لكل طالب أو طالبة :

- ١ - متغير ولنطلق عليه اسم ISEX يأخذ القيمة 1 أو 2 ليعين للحاسب النوع ( طالب أم طالبة ) ،  
٢ - متغير ولنطلق عليه اسم KCLASS يأخذ القيمة من 1 حتى 6 ليعين للحاسب الدرجة التي حصل عليها ذلك الطالب ( أو الطالبة ) .

ولذا يمكننا أن نتخيل مخطط التدفق لحل تلك المشكلة كالتالي :





شكل (٥-٢) مخطط التدفق للمثال أعلاه

وبذا يمكن كتابة البرنامج كالتالي :

```

C      PROGRAM 1 TO CALCULATE THE NO. OF STUDENTS,
C      (MALES, FEMALES)
      NMALES = 0
      NFEMAL = 0
      JRADE 1 = 0
      JRADE 2 = 0
      JRADE 3 = 0
      JRADE 4 = 0
      JRADE 5 = 0
      JRADE 6 = 0
      READ (7, 100) N
100    FORMAT (18)

      KOUNT = 1
      READ (7,5) ISEX, KLASS
      FORMAT (2I3)
      GOTO (10, 15), ISEX
10     NMALES = NMALES + 1
      GOTO 20
15     NFEMAL = NFEMAL + 1
      GOTO (25,30, 35,40, 45, 50), KLASS
20     JRADE 1 = JRADE 1 + 1
      GOTO 55
30     JRADE2 = JRADE2 + 1
      GOTO 55
35     JRADE3 = JRADE3 + 1
      GOTO 55
40     JRADE4 = JRADE4 + 1
      GOTO 55
45     JRADE5 = JRADE5 + 1
      GOTO 55
50     JRADE6 = JRADE6 + 1
      IF (KO UNT. EQ. N) GOTO 60
      KOUNT = KOUNT + 1
      GOTO 1
60     WRITE (7,65) NMALES, NFEMAL, JRADE1, JRADE2,
1     JRADE3, JRADE4, JRADE5, JRADE6
65     FORMAT (8I6)
      STOP
      END

```

وقد يتساءل البعض عن امكانية تنفيذ البرنامج السابق باستخدام تعليمة اذا .. IF المنطقية ، وفي هذه الحالة ماهو الفرق بين الطريقتين ؟ والاجابة على ذلك نعم كما على :

C	<pre> PROGRAM 2 OF EXAMPLES - 2-3 NMALES = 0 . . . JRADE6 = 0 . . . KOUNT = 1  1 READ (7,5) ISEX, KCLASS 5 FORMAT (2I3)   IF (ISEX. EQ. 1) GOTO 10   NFEMAL = NFEMAL + 1   GOTO 15 10 NMALES = NMALES + 1 15 IF (KCLASS. EQ. 1) GOTO 20   IF (KCLASS. EQ. 2) GOTO 25   IF (KCLASS. EQ. 3) GOTO 30   IF (KCLASS. EQ. 4) GOTO 35   IF (KCLASS. EQ. 5) GOTO 40   JRADE6 = JRADE6 + 1   GOTO 55 20 JRADE1 = JRADE1 + 1   GOTO 55 25 JRADE2 = JRADE2 + 1   GOTO 55 30 JRADE3 = JRADE3 + 1   GOTO 55 35 JRADE4 = JRADE4 + 1   GOTO 55 40 JRADE5 = JRADE5 + 1 55 IF (KOUNT. EQ. N) GOTO 60    . . .    . . .    . . . END </pre>	<p>كما في برنامج 1</p> <p>كما في برنامج 1</p>
---	---	---

وقد يتبين مما سبق أننا إستبدلنا تعليمة الانتقال GOTO المحسوبة بمجموعة متتالية من تعليمة اذا .. IF المنطقية ، وبالطبع ستكون نتائج البرنامجين متطابقة ، ولكن قد يختلف زمن تنفيذ كل برنامج عن الآخر حيث يتوقف وقت تنفيذ أي برنامج على التعليمات النهائية الخاصة بلغة الحاسب . (Machine Language)

تقريبات على تعليمات الانتقال **GO TO** ، اذا **IF** .

١ - أوجد القيمة النهائية للمتغير  $M$  التي نحصل عليها بعد الانتهاء من تنفيذ اجزاء البرامج التالية ، اذا كانت قيمة  $M$  في البداية هي 3 وقيمة  $N$  هي 6 :

i) 

IF (3*M. EQ. N) M=M+2
M = M+3

ii) 

IF (N. GT. M) GO TO 30
M = M+1
GO TO 20
30 M = N
20 M = M - N

iii) 

IF (N - M) 3,6,3
6 M = M+1
3 M = N

iv) 

IF (3*M - 2*N) 2,5,5
2 M = N
5 M = M+1

v) 

IF (2*M. LT. N) M=M-2
M = M+2

vi) 

IF (3*N. LE. 2*M) GO TO 10
M = M+1
GO TO 20
10 M = N
20 M = M+N

٢ - أوجد تعليمات الانتقال المحسوبة الغير صحيحة في التعليمات التالية ، وبين سبب الخطأ .

- i) GO TO (10,5,8,0,15), L
- ii) GO TO (7,11,11,11,9,7), K
- iii) GO TO (4,5,9), Y
- iv) GO TO (8,10,3,6) M

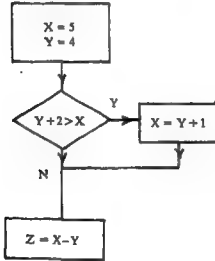
٣ - اكتب تعليمة اذا IF الحسابية التي تكافئ التعليمة التالية :

- i) GO TO (15,10,30), L
- ii) GO TO (22,55,33,77,55), KOKO

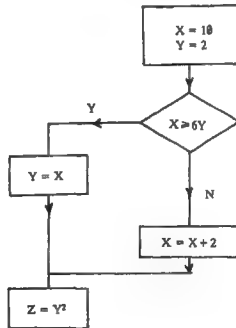
٤ - في تعليمات اذا IF التالية توجد أخطاء اما في التعليمة نفسها أو عند تنفيذها ، صحح تلك الأخطاء :

- i) IF (I=N), 4,10,5
- ii) IF (X+Y-4) 5,5,7
- iii) IF X-0 4,5,6
- iv) IF (X+Y\*Y) 5,2
- v) 10 || IF (X+Y) 5,5,10
- vi) 16 || IF (A-(B+C) 3, 10,50
- vii) IF (X+4) 3,7
- viii) 3 || IF (A\*N) 8,3,6

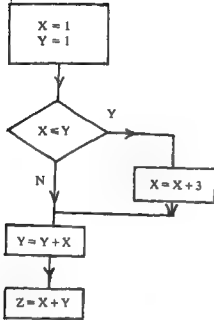
٥ - في مخططات التدفق التالية ، اكتب اجزاء البرامج المناسبة لكل منها ، وأوجد قيمة Z الناتجة :



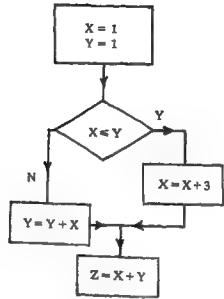
i



ii



iii



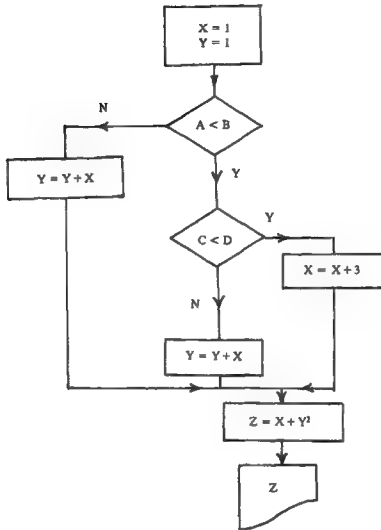
iv

٦ - عندما تكون قيمة المتغير  $X$  بين صفر وواحد فإن قيمة المتغير  $Y$  تساوي  $X^2$  ، عندما تكون قيمة المتغير  $X$  بين ثلاثة وأربعة فإن قيمة المتغير  $Y$  تساوي  $(-X)$  ، إذا كانت قيمة  $X$  غير ذلك فإن  $Y$  تساوي صفراً .

ارسم مخطط تدفق لتلك المشكلة وأكتب الجزء من البرنامج المناسب .

٧ - اكتب الجزء من البرنامج الذي يناسب مخطط التدفق التالي شكل (٥-٤) ، وأحسب قيمة  $Z$  في الحالات التالية :

- 1)  $A = 2$  ,  $B = 3$  ,  $C = 3$  ,  $D = 2$
- 2)  $A = 2$  ,  $B = 3$  ,  $C = 2$  ,  $D = 3$
- 3)  $A = 3$  ,  $B = 2$  ,  $C = 2$  ,  $D = 3$
- 4)  $A = 3$  ,  $B = 2$  ,  $C = 3$  ,  $D = 2$



شكل (٤-٥)

٩ - أرسم مخطط تدفق واكتب الجزء من البرنامج لحساب :

$$S = \sum_{i=1}^N X_i Y_i$$

- ١٠ - من المعروف نظرياً أن مجموع أي ضلعين في مثلث أكبر من الضلع الثالث فإذا أعطيت ٦٠ قيمة متتالية ، كل ثلاث منها تمثل أطوالاً لأضلاع قد تكون مثلثاً أو لا تكون . اكتب برنامج يقرأ تلك القيم كل ثلاثة على انفراد ثم يبين إن كانت تلك القيم تمثل مثلثاً أم لا .  
( ملحوظة ) A , B , C تكون مثلثاً إذا كان  $B + C > A; A + C > B; A + B > C$

- ١١ - من المعروف نظرياً أن جذري معادلة من الدرجة الثانية :  $AX^2 + BX + C = 0$  يمكن حسابها من القانون :

$$X_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A} ; \quad X_2 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

بحيث أنه إذا كانت :

- (أ)  $(B^2 - 4AC)$  أكبر من الصفر ، فإن الجذرين يكونان حقيقيين .  
 (ب)  $(B^2 - 4AC)$  أقل من الصفر ، فإن الجذرين يكونان تخيليين .  
 (جـ)  $(B^2 - 4AC)$  تساوي صفراً ، فإن الجذرين يكونان متساويين وكل منهما يساوي :  $-\frac{B}{2A}$   
 (د)  $A$  تساوي صفراً فإنه لا توجد جذور للمعادلة .  
 أرسم مخطط تدفق ، ثم أكتب الجزء من البرنامج المناسب .

١٢ - اكتب برنامج لحساب :

$$S = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \dots + \frac{99}{100}$$

١٣ - اكتب برنامج لحساب :

$$P = \frac{1}{1^2} \cdot \frac{3}{2^2} \cdot \frac{5}{3^2} \cdot \dots \cdot \frac{2N-1}{N^2}$$

١٤ - اكتب برنامج لحساب :

$$S = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots \pm \frac{1}{N}$$

- ١٥ - في أحد المحال التجارية ، يتم عمل خصم على المشتريات طبقاً للشروط التالية :
- (أ) إذا كانت قيمة المشتريات أقل من ٢٠٠ ريالاً ، فلن يكون هناك خصم ،  
 (ب) إذا كانت قيمة المشتريات أكبر من أو يساوي ٢٠٠ ريالاً وأقل من أو يساوي ٥٠٠ ريالاً فإن قيمة الخصم تساوي ١٠٪ من قيمة المشتريات ،  
 (جـ) إذا كانت قيمة المشتريات أكبر من ٥٠٠ ريالاً ، فإن قيمة الخصم تساوي (٨٠٪) من المتبقي من قيمة المشتريات بعد خصم ٥٠٠ ريالاً منها .  
 أرسم مخطط تدفق وأكتب البرنامج المناسب .

- ١٦ - في إحدى الجامعات تم عمل احصاء للطلبة والطالبات المنتحقين بها ، وذلك بعمل بطاقة لكل طالب أو طالبة مرقب بها البيانات التالية :



السن - النوع ( ١ للطالبة ، ٢ للطلاب ) - المستوى الدراسي ( ١ : للمستوى الأول ، ٢ : للمستوى الثاني ، ٥ ، ٥٠٠ : للمستوى الخامس ، ٦ : لغير المقيد بنظامياً ) - الحالة العائلية ( ١ : للأعرب ، ٢ : للمتزوج ) .

يتم الكشف عن نهاية البطاقات بوضع بطاقة بها قيمة تساوي ٣ في خانة النوع .

أرسم مخطط تدفق واكتب البرنامج المناسب لحساب :

- ( أ ) النسبة المئوية للطلبة ،
- ( ب ) النسبة المئوية للطالبات ،
- ( ج ) النسبة المئوية للطلبة والطالبات في المستويات الخمس المختلفة .
- ( د ) النسبة المئوية للطلبة والطالبات الذين تزيد أعمارهم عن ٣٠ سنة .
- ( هـ ) النسبة المئوية للمتزوجون .

١٧- اكتب برنامج يقرأ ثلاث قيم صحيحة ويقارن بينها ، بحيث اذا :

- ( أ ) لم تكن هناك قيمتين متساويتين ، يكتب الرقم ٥
- ( ب ) كانت هناك قيمتين متساويتين ، يكتب ترتيبهما ( الأول والثالث مثلاً يكتب 1-3 ) .

( ج ) كانت القيم الثلاث متساوية ، يكتب 3 .

أختبر البرنامج على الحالات التالية : 666 , 133 , 979 , 998 , 787

١٨- مجموعة من البطاقات مثقبة في كل منها رقماً صحيحاً موجباً ( أقل من ١٠٠٠ ) ، وفي نهاية

تلك المجموعة من البطاقات بطاقة لتعريف الحاسب بنهاية المجموعة .

( أ ) أكتب برنامجاً لايجاد أكبر رقم في تلك المجموعة .

( ب ) أكتب برنامجاً لايجاد أصغر رقم في تلك المجموعة .

١٩- بافتراض أن تعداد السكان في الدولتين أ ، ب هما ٥٢ ، ٨٥ مليون نسمة على التوالي ، وكان

معدل نمو السكان بهما هي ٤% ، ٢% على التوالي . أكتب برنامجاً لأعطاء تعداد السكان

( لأقرب ١٠٠٠ ) في كل من الدولتين كل سنة أن يزيد تعداد السكان في الدولة أ عن الدولة

ب . وأوجد عدد السنوات التي يتحقق فيها ذلك .



## الفصل السادس

### تعلیمة حلقة التنفيذ المتكرر



## الفصل السادس

### DO loop

### تعلمية حلقة التنفيذ المتكرر

“Every Program can be shortened”

مقدمة :

أي « كل برنامج يمكن تصغيره أو إختصاره » ، شعار ظل ماثلاً ولن ينسى منذ بدأنا خطواتنا الأولى في تعلم لغة الفورتران ، وبالطبع فهذا الشعار لا يخص مستملي لغة الفورتران فقط ولكن يمكن تطبيقه على أي لغة أخرى من لغات الحاسب ولكن العبرة هو في تطبيق الشعار الثاني “THINK” أي فكر وهو أحد الشعارات وأهمها التي تضعها شركات IBM على مطبوعاتها وملصقاتها .

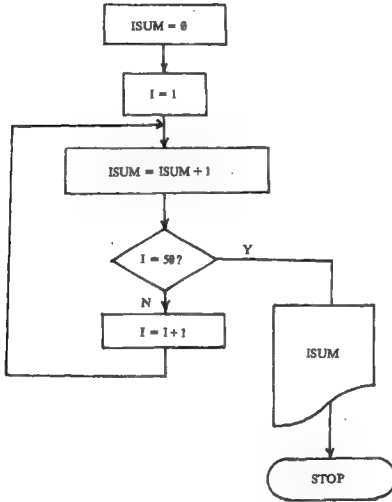
فأي مشكلة حسابية يمكن حلها كتابة أكثر من برنامج وجميعها ستعطي نتائج أن لم تكن متفقة تماماً مع بعضها ، فستكون متقاربة(\*) وهناك مشاكل خاصة تتطلب دقة كبيرة في حساباتها مثل المشاكل التي تتعلق بتجارب معملية فيزيائية أو كيميائية أو فلكية أو ... الخ ، ولكن معظم المشاكل لا تتطلب تلك الدقة الكبيرة في نتائجها ، فلا أقل من أن نراعي في حلها السهولة والبساطة والاختصار خاصة في كتابة برنامج ذلك الحل .

وأحدى الوسائل الفعالة في اختصار معظم البرامج المكتوبة بلغة الفورتران هو إستخدام حلقات التنفيذ المتكررة ، والتي تختص بتنفيذ مجموعة تعليمات معينة في البرنامج ، عدد معين من المرات بصفة تكرارية . ولنفترض المثال البسيط التالي والذي يراد فيه بإستخدام تعليمات الفورتران التي سبق دراستها حتى الآن ، إيجاد مجموع الأعداد التي تقع بين العددين ١ ، ٥٠ أي أن المطلوب إيجاد قيمة :

$$\sum_{i=1}^{50} i = 1 + 2 + 3 + \dots + 50$$

- (هـ) قد تكون هناك فوارق في نتائج حل مشكلة ما ، نظراً لما تتعرض له الحسابات من نوعين مختلفين من الأخطاء :
- (أ) أخطاء حذف التقريب الدائري Round off error : وهو بأن تقوم بحذف الأرقام ذات الأهمية الصغرى في النتائج الحسابية ، ثم تقرب النتيجة الى عدد معين من الأرقام العشرية .
- (ب) أخطاء الأنهاء المبكر Truncation errors : وهو بأن تقوم بإنهاء العمليات الحسابية بأخذ حدود معينة وإهمال بقية تلك الحدود توفيراً للوقت والجهد . فالتقدير ...  $0.33 \neq 0.3 \neq \frac{1}{3}$

فأذا افترضنا أن نتيجة الجمع سيتم تخزينها في المتغير ISUM ، فأذا يمكننا إيجاد حاصل الجمع طبقاً  
لخطة تدفق البيانات التالي :



شكل (١-٦)

ويكون البرنامج كالتالي :

	ISUM = 0 .....	وضع قيمة المتغير الذي سيتم فيه التجميع بالقيمة صفر
	I = 1 .....	عداد سيأخذ القيم 1, 2, 3, ... 50
5	ISUM = ISUM + I .....	القيمة الجديدة للمتغير ISUM تساوي القيمة السابقة لنفس المتغير مضافاً إليها قيمة المتغير I
	IF (I.EQ.50) GOTO 10	
	I = I + 1 .....	طالما أن قيمة المتغير I لم تصل بعد إلى القيمة 50 فسيؤيد قيمة ذلك المتغير بواحد صحيح ....
	GOTO 5 .....	الانتقال إلى التعليمة رقم 5 في البرنامج وتكرار تنفيذ الخطوات التي تليها ....
10	WRITE (6,15) ISUM .....	كتابة قيمة المتغير الذي تمت فيه عملية التجميع
15	FORMAT (I6)	
	STOP	
	END	

نلاحظ في المثال السابق مايلي :

١ - أن مجموعة التعليمات التي تبدأ بالتعليمة رقم 5 وتنتهي بالتعليمة GOTO 5 تكون فيما بينها حلقة (loop) ، يتكرر تنفيذها عدداً معيناً من المرات .

٢ - يتوقف تكرار تنفيذ تلك الحلقة على قيمة المتغير I (العداد) ، فإن كانت قيمة I لا تساوي 50 ، أي لم يصل إلى القيمة النهائية المطلوب الحساب عندها فإن الحاسب سيقوم بتنفيذ تلك الحلقة مرة أخرى وهكذا حتى تصل قيمة المتغير I إلى القيمة النهائية المطلوب التوقف عندها .

سؤال : ماذا يحدث إذا استبدلنا التعليمة رقم 5 بالتعليمة التي تليها أي بدلنا هاتين التعليمتين بحيث يصير البرنامج السابق كالتالي :

	ISUM = 0
	I = 1
5	IF (I.EQ.50)GOTO 10
	ISUM = ISUM + I
	I = I + 1
	GOTO 5
10	WRITE (6,15) ISUM
15	FORMAT (I6)
	STOP
	END

هل سنصل الى نفس النتيجة ؟ واذا كان هناك اختلافا في النتيجة فما هو السبب وكيف يمكن تصحيحه ؟

٣ - أن المتغير I يبدأ بقيمة معينة يمكن أن نسميها القيمة الابتدائية Initial value or starting value أي التي سيبدأ العد بها ، وينتهي بقيمة معينة يمكن أن نسميها القيمة النهائية Final value .

٤ - أن المتغير I والذي يمثل عددا في هذا المثال ، يستخدم أيضاً كمتغير يمكن استخدامه داخل بعض التعليمات التي تحتوي عليها الحلقة (  $ISUM = ISUM + I$  ) .

٥ - عند تكرار تنفيذ تعليمات تلك الحلقة فإن المتغير I يزداد في كل مرة بمقدار الوحدة ولكن زيادة قيمة المتغير I تم عن طريق تعليمة داخل الحلقة نفسها (  $I = I + 1$  ) ولذا فيمكن تغيير تلك التعليمة بحيث يزداد المتغير I بأي مقدار صحيح موجب .

( سؤال : هل يمكن تحويل البرنامج السابق بحيث يقوم بجمع قيم الأعداد الفردية الواقعة بين 1 , 50 ؟ )

٦ - طالما أن المتغير I يستخدم كعداد ليحسب عدد مرات تنفيذ التعليمات الموجودة داخل تلك الحلقة ، فيجب أن نتوقع أن :

( أ ) أول قيمة سيأخذها المتغير I ستكون أقل من القيمة النهائية .. واذا تساوت القيمتين فإن الحاسب سيقوم بتنفيذ التعليمات الموجودة داخل الحلقة مرة واحدة فقط .  
( ب ) أن المتغير I سيزداد بقيم صحيحة موجبة .

في المثال البسيط السابق ، عرفنا كيف يمكن للحاسب أن يقوم بتنفيذ مجموعة من التعليمات تكون فيما بينها حلقة loop . وفي الحقيقة فكثيراً ما نواجه بعض المشكلات التي يتطلب تنفيذها على الحاسب عمل برامج تحوي على مثل تلك الحلقات المتكررة ، وفي كل حلقة منها لابد من تبليغ الحاسب بزيادة مقدار المتغير الذي يمثل العداد بقيمة ما ، وفي كل مرة عند تنفيذ كل حلقة لابد من أن نجعل الحاسب يسأل عن قيمة المتغير الذي يمثل العداد وهل وصل الى القيمة النهائية المطلوبة أم لا ؟ وبالطبع فأن مثل تلك التعليمات المثلثة في زيادة قيمة المتغير الذي يمثل العداد وكذلك في عمل مقارنة في كل مرة بين القيمة التي وصل اليها العداد والقيمة النهائية وكذلك في عملية الانتقال الى بداية الحلقة في كل مرة يتكرر فيها تنفيذ مجموعة التعليمات التي نحبها ، تستغرق وقتاً كبيراً نسبياً من الحاسب حيث سيتوقف هذا الوقت على عدد مرات تنفيذ التعليمات التي نحبها الحلقة وعلى عدد التعليمات التي توجد داخل تلك الحلقة ، كما ستزيد من حجم البرنامج نفسه .

ولذا فإن لغة الفورتران تحوي تعليمة خاصة وهامة يمكن ويستحسن إستخدامها عندما يتطلب الأمر تنفيذ مجموعة التعليمات بصفة تكرارية لعدد معين من المرات ، وهذه التعليمة هي :



## DO Statement (loop)

## ١-٦ الصورة العامة لتعليمة حلقة التنفيذ المتكرر

وتكتب في الصورة العامة كالتالي :

vvvvv	DO n i = i <sub>i</sub> , i <sub>f</sub> , i <sub>g</sub>
-------	---

حيث :

vvvvv : رقم بدء تعليمة حلقة التنفيذ المتكررة ، وهو رقم إختياري يتوقف وجوده أو عدم وجوده على واضح البرنامج .

n : رقم نهاية تعليمة حلقة التنفيذ المتكرر ، ولا بد من تواجده في البرنامج .

i : اسم متغير صحيح Integer variable يستخدم كعداد في حلقة التنفيذ المتكرر وقد يستخدم كمتغير صحيح داخل بعض تعليمات الحلقة نفسها ، ويطلق عليه اسم Index .

i<sub>i</sub> : القيمة الابتدائية ( أول قيمة Initial value ) للمتغير i .

i<sub>f</sub> : القيمة النهائية ( آخر قيمة Final Value ) للمتغير i .

i<sub>g</sub> : القيمة التي سيزداد بها المتغير i في بدء تكرار تنفيذ مجموعة التعليمات التي تحويها الحلقة . ويطلق عليها اسم ( الخطوة Increment or Step ) .

والقيم الثلاث i<sub>i</sub>, i<sub>f</sub>, i<sub>g</sub> لا بد أن تكون قيماً عديدة صحيحة موجبة أو أسماء متغيرات صحيحة مسبقاً اعطائها قيماً عديدة صحيحة موجبة في البرنامج .


وإذا كان المتغير الصحيح i يزداد بمقدار الوحدة ( أي أن i<sub>g</sub> = 1 ) فإن الصورة العامة تكون كالتالي :

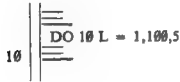
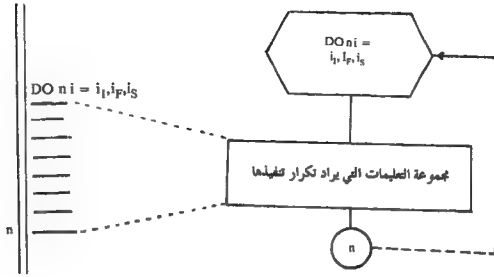
vvvvv	DO n i = i <sub>i</sub> , i <sub>f</sub>
-------	--

وفي كلتا صورتين يجب :

١ - وضع اسم المتغير الصحيح i على يسار علامة التساوي ، بينما يوضع على يمينها قيمة i<sub>i</sub> أولاً ، ثم قيمة i<sub>f</sub> ثانياً ، ثم قيمة i<sub>g</sub> ( أن تطلب الأمر ذلك ) . كما يجب التأكد من وجود الفاصلة Comma بين كل قيمة وأخرى .

٢ - أن تأتي التعليمة التي رقمها n والتي تمثل تعليمة نهاية حلقة التنفيذ بعد ( وليس قبل ) بدء تعليمة حلقة التنفيذ المتكرر .

ومثل حلقات التنفيذ في مخططات التدفق بالرمز  وبهذا يمكن رسم مخطط التدفق لتعليمة حلقة التنفيذ المتكرر كالتالي :



#### مثال ٦-٩-١ :

- في هذا المثال سنجد أن الحاسب يقوم بتنفيذ مجموعة التعليمات التي تمثل حلقة التنفيذ المتكرر كالتالي :
- يأخذ قيمة المتغير  $L = 1$  وينفذ مجموعة التعليمات التي تلي تعليمة حلقة التنفيذ المتكرر DO إلى أن يصل إلى التعليمة التي رقمها 10 والتي تمثل آخر تعليمة في الحلقة .
  - ثم يأخذ قيمة المتغير  $L = 1 + 5 = 6$  وينفذ مجموعة التعليمات حتى يصل إلى التعليمة 10 .
  - يأخذ قيمة المتغير  $L = 6 + 5 = 11$  بعد ذلك ويستمر كما سبق إلى أن يصل إلى قيمة L النهائية والتي لا بد أن تكون أقل من أو تساوي القيمة النهائية 100 ، أي إلى أن يصل إلى قيمة :  $L = 91 + 5 = 96$ .

نلاحظ في هذا المثال :

- عند مقارنة تعليمة حلقة التنفيذ في هذا المثال بالصورة العامة سنجد أن :  $n = 10, i = L, i_f = 1, i_p = 100, i_g = 5$

- إن المتغير  $L$  والذي يمثل العداد في هذا المثال لن يتزايد بمقدار الوحدة عند تكرار تنفيذ التعليمات التي تمثل حلقة التنفيذ بل سيأخذ القيم :  
 $1, 1+5 = 6, 6+5 = 11, 11+5 = 16, 16+5 = 21, \dots, 91+5 = 96$

وهذا ماكننا نعبر عنه في السابق بالتعليمة :  $L = L+5$  .

- يستمر الحاسب في تكرار تنفيذ التعليمات التي تحويها حلقة التنفيذ المتكرر ولا يخرج من هذه الحلقة إلا بعد أن تصل قيمة المتغير  $L$  إلى 101 ( $96+5$ ) أي عندما تكون قيمة  $L$  أكبر من القيمة النهائية  $96$  .

مثال ٦-٩-٢ :

$$\begin{array}{l|l} & DO \ 5 \ K = 3,7 \\ 3 & \equiv \end{array}$$

نلاحظ في هذا المثال :

- ١ - أن أول قيمة للمتغير  $K$  ستكون 3 وآخر قيمة هي 7 .
- ٢ - أن تعليمة حلقة التنفيذ  $DO$  اقتصر على المتغيرين  $i$  و  $j$  ، ولذا فإن الحاسب سيعتبر قيمة المتغير  $i$  هي 1 . وبالتالي فإن قيم  $K$  ستكون 3, 4, 5, 6, 7 على الترتيب .

مثال ٦-٩-٣ :

$$\begin{array}{l|l} & IDRIS = 5 \\ & MORAD = 60 \\ & KAMEL = 10 \\ 7 & DO \ 7 \ KOUNT = IDRIS, MORAD, KAMEL \\ & \equiv \end{array}$$

وفي هذا المثال نجد أن :

- ١ - تعليمة حلقة التنفيذ المتكرر تحتوي على أسماء لمتغيرات صحيحة وليست قيماً صحيحة ، فالمتغير  $i$  في الصورة العامة والذي يمثل العداد في تلك التعليمة إستبدل بالمتغير  $KOUNT$  ، والمتغير الصحيح  $i$  بالمتغير  $IDRIS$  ، والمتغير الصحيح  $j$  بالمتغير  $MORAD$  ، والمتغير الصحيح  $k$  بالمتغير  $KAMEL$  .

٢ - لابد من تعريف الحاسب مسبقاً بقيم المتغيرات الصحيحة KAMEL, MORAD, IDRIS  
مراعاة أن قيمة المتغير الصحيح IF ستكون أكبر من أو تساوي قيمة المتغير الصحيح IA وأن قيم  
المتغيرات الصحيحة الثلاثة لابد أن تكون موجبة .

ملاحظة :

ليس من الضروري أن تكون المتغيرات الصحيحة IA, IF, ID جميعها قيماً صحيحة موجبة أو أسماء  
للمتغيرات صحيحة ولكن قد تحوي تعليمة حلقة التنفيذ المتكرر خليط من القيم والمتغيرات الصحيحة

على سبيل المثال :

```

9 | ISTART = 10
  | ISTEP = 5
  | DO 9 L = ISTART,100,ISTEP
  | ---
  | ---
  | ---
  | ---
  | ---

```

مثال ٦-١-٤ :

أوجد قيمة M التي سيكتبها الحاسب من البرنامج التالي :

```

10 | M = -50
   | DO 10 K = 5,20,5
   | M = M + K
   | WRITE (7,15) M
15 | FORMAT ('MM=', 13)

```

لاحظ استخدام العدد K كمتغير صحيح في حساب قيم M في التعليمة رقم 10 .

- في البداية فإن قيمة M هي -50 .

- عندما يبدأ الحاسب في تنفيذ تعليمة حلقة التنفيذ المتكرر DO سيكون التنفيذ كالتالي :

المتكرر رقم	قيمة K	قيمة M
1	5	$-50 + 5 = -45$
2	10	$-45 + 10 = 35$
3	15	$-35 + 15 = -20$
4	20	$-20 + 20 = 0$

- عند وصول قيمة المتغير K الى قيمته النهائية ( 20 في مثالنا هذا ) ، يبدأ الحاسب في الخروج من حلقة التنفيذ المتكرر ليكمل تنفيذ بقية تعليمات البرنامج أي كتابة قيمة M . وبذا ستوقع من الحاسب أن يكتب النتيجة التالية :  $VM = 000$

( سؤال : حاول كتابة البرنامج السابق بدون إستخدام تعليمة حلقة التنفيذ المتكرر ) .

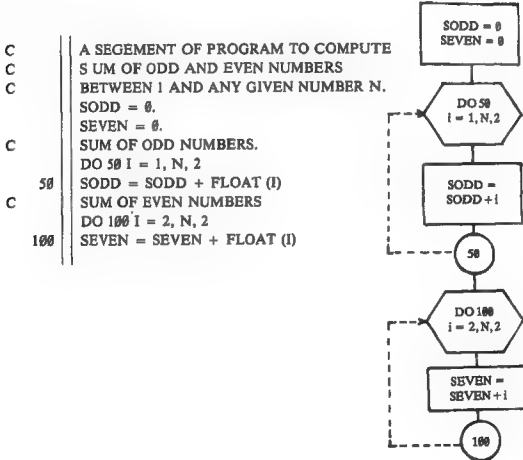
مثال ٦-١-٥ :

اكتب برنامج لحساب مجموع الأعداد الفردية وكذلك مجموع الأعداد الزوجية الواقعة بين العددين 1 , N .

لتفرض أن المتغير الذي سيحتوي على مجموع الأعداد الفردية هو SODD .  
وأن المتغير الذي سيحتوي على مجموع الأعداد الزوجية هو SEVEN .

وحيث أنه سيم فيها عمليات تجميع ، فلا بد أولاً من تصفيرهما ، أي جعل :

$SEVEN = 0$  ;  $SODD = 0$ .



نلاحظ في هذا المثال :

١ - أن هناك حلقتي تنفيذ أحدهما لحساب SODD والأخرى لحساب SEVEN وفي الأثنين كان المتغير الصحيح I ( الذي يمثل العدد ) واحدا لم يتغير . وفي هذا الاجراء نوع من توفير خلايا التخزين في الحاسب . ولن يكون هناك تداخل بين قيمة المتغير الصحيح I في حلقة التنفيذ الأولى وقيمه في حلقة التنفيذ الثانية ، اذ أنه بمجرد الانتهاء من تنفيذ الأولى ستكون قيمة I هي N أو ( N-1 ) اذا كانت N زوجية . وعند البدء في تنفيذ حلقة التنفيذ الثانية سيأخذ المتغير الصحيح I القيمة الابتدائية I في تلك الحلقة وهي 2 ، ولن يستطيع الحاسب عمل ذلك الا بعد الانتهاء تماماً من تنفيذ حلقة التنفيذ الأولى .

٢ - أهمية المتغير الصحيح I وارتباطه بالمتغير الصحيح I ، حيث أمكن عن طريق هذا الارتباط الحصول على الأعداد الفردية والزوجية مباشرة .

#### CONTINUE Statement

#### ٦-٢ تعليمية الاستمرار أو المواصلة

وكما يتضح من إسم تلك التعليمية وهو الاستمرار أو المواصلة ، ولذا فإن الحاسب عندما يصل إليها في البرنامج لتنفيذها فلن يكون عليه إلا أن يستمر في تنفيذ تعليمات البرنامج طبقاً لما يقوم به من حسابات ، ولذا فهي تختلف عن بقية التعليمات التي سبق شرحها والتي كان لكل منها هدف تعمل في البرنامج . ولذا أيضاً يطلق على مثل هذه الأنواع من التعليمات بالتعليمات الخادعة أو الزائفة Dummy Statements .

ومع ذلك ففي بعض المواضع من البرنامج أحياناً ، لانستطيع الاستغناء عنها ، بل تصبح ضرورية الاستخدام كما سيبين من الأمثلة التالية . وغالباً ما تكون هي التعليمية النهائية لحلقة تنفيذ متكررة ، خاصة اذا كانت تلك الحلقة التنفيذية تحوي بداخلها تعليمية اذا IF .

مثال ٦-٢-١ :

```

10  X = 5.
    CONTINUE
15  X = X + 3.

```

في هذا المثال عندما يصل الحاسب الى تنفيذ التعليمية التي رقمها 10 فإنه سيخصص القيمة 5 للمتغير X . بعد ذلك ينزل الى التعليمية التالية والتي تبلغ الحاسب باستمرار التنفيذ ، وبالتالي فليس

أمام الحاسب سوى الاستمرار في تنفيذ تعليمات البرنامج ولذا يبدأ الحاسب في تنفيذ التعليمة التي رقمها 15 والتي تجعله يخصص القيمة 8 للمتغير X .

ومن الواضح أن وضع تعليمة الاستمرار في مثل تلك الظروف ليس لها داع لأنها لن تغير من حسابات البرنامج ، كما أنها لن تغير من سير تنفيذ تعليمات البرنامج ولذا يمكن الاستغناء عنها للمساعدة في جعل البرنامج قصيراً .

مثال ٦-٢-٧ :

```

10  |=====
    | X = 0.
    | DO 10 I = 1, 3
    | X = X + 3.
    | CONTINUE
    | Y = X * X
    |=====

```

في هذا المثال نجد أن آخر تعليمة تنفيذية في حلقة التنفيذ المتكرر DO هي تعليمة الاستمرار ، وبالتالي فإن الحاسب في كل مرة يصل فيها إلى تلك التعليمة فما عليه إلا أن يغير قيمة I بحسب قيمة X الجديدة . بعد تكرار تلك العملية ثلاث مرات يبدأ الحاسب في تنفيذ التعليمة الخاصة بحساب المتغير Y . ولذلك فإن المتغير X سيأخذ القيم 3, 6, 9 خلال تنفيذ حلقة التنفيذ المتكرر ، ثم بعد ذلك يتم حساب قيمة المتغير Y والذي يساوي 81 .

ولذلك فإن وضع تعليمة الاستمرار هنا لالزوم لها أيضاً ، لأننا لو كتبنا البرنامج بدونها فلن تتغير قيم X أو Y . ويكون البرنامج كالتالي :

```

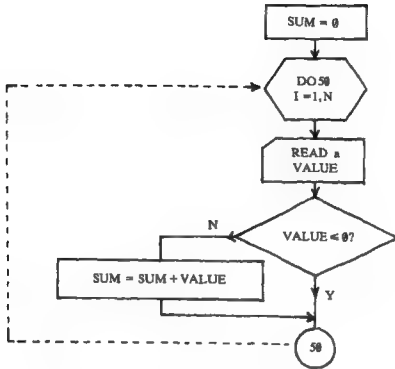
10  |=====
    | X = 0.
    | DO 10 I = 1,3
    | X = X + 3.
    | Y = X * X
    |=====

```

مثال ٦-٢-٨ :

مجموعة من القيم عددها N بعضها موجب والآخر سالب ، وبعد أن يتم قراءة كل قيمة على حدة يتم جمع القيم الموجبة فقط منها ، لإكتب برنامجاً لتحقيق ذلك .

نفترض أن كل قيمة سيتم قراءتها عن طريق المتغير VALUE ، كما أن مجموع القيم الموجبة سيخزن في المتغير SUM ، لذا يمكن رسم مخطط التدفق كالتالي :



شكل (٤-٦)

C     A PROGRAM TO CALCULATE SUM OF + VE NUMBERS  
       SUM = 0.  
       DO 50 I = 1, N  
       READ (5,5) VALUE  
       IF (VALUE. LE. 0.) GO TO 50  
       SUM = SUM + VALUE  
 50    CONTINUE

في هذه المشكلة تتضح أهمية تعليمة الاستمرار ، حيث أنه لا يوجد بديل لها لتكون آخر تعليمة تنفيذية في حلقة التنفيذ المتكرر DO فبفضلها أمكننا التهرب من جميع القيم السالبة في المتغير SUM والذي كان لابد سيحدث اذا لم توجد تعليمة الاستمرار في هذا الموضع ، وكانت التعليمة التي رقمها 50 ، أي آخر تعليمة في حلقة التنفيذ هي :

50    SUM = SUM + VALUE

ففي تلك الحالة فإن المتغير VALUE كان سيم جمعه على المتغير SUM سواء كان سالبا أم موجبا .



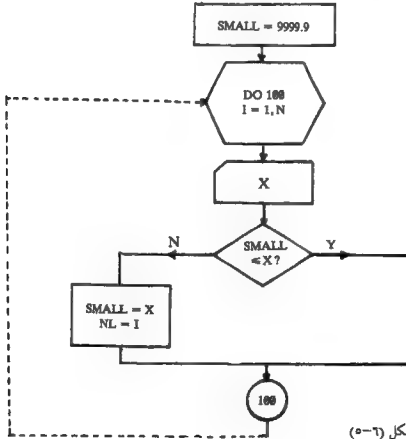
## مثال ٦-٢-٤ :

مجموعة من الأعداد الموجبة عددها  $N$  ، ادخلت الى الحاسب بطريقة عشوائية ( أي ليست بطريقة تصاعدية للأعداد أو تنازلية ) . إكتب برنامج لتعيين أصغر قيمة في هذه الأعداد وترتيبه في مجموعة تلك الأعداد .

لايجاد قيمة أصغر عدد في مجموعة أعداد ، سنبدأ بإفتراض أن أصغر عدد SMALL هو عدد له قيمة كبيرة جداً أو كبير نسبياً عن أي عدد في مجموعة الأعداد ( كما يمكن أن تكون قيمة المتغير SMALL هو قيمة أول عدد في المجموعة ) . ونبدأ في مقارنة المتغير SMALL بمجموعة الأعداد ، وكلما وجدنا عدداً أصغر من SMALL نخفظ به ونخصص قيمته الى المتغير SMALL ونحفظ بترتيبه في مجموعة الأعداد وهكذا .

ولايجاد أكبر عدد ، يمكن إتباع نفس الخطوات بإفتراض عدد صغير جداً في البداية ثم حجز قيمة وترتيب أي عدد له قيمة أكبر من قيمة ذلك العدد .

ويمكن تصور مخطط التدفق كالتالي :



شكل (٥-٦)

ويكون الجزء من البرنامج لحل تلك المشكلة كالتالي :

C	A SEGEMENT OF PROGRAM TO FIND THE SMALLEST NUMBER.
	SMALL = 9999.9
	DO 100 I = 1, N
	READ (5, 30) X
30	FORMAT (F 8.2)
	IF (SMALL. LE. X) GO TO 100
	SMALL = X ..... حجز قيمة العدد الأصغر في المتغير SMALL
	NL = I ..... حجز ترتيب العدد الأصغر في المتغير NL
100	CONTINUE

### Nested DO Loops

### ٣-٦ حلقات التنفيذ المتداخلة :

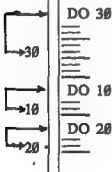
#### مقدمة :

ماذا يقصد بحلقات التنفيذ المتداخلة ؟

علمنا فيما سبق معنى حلقة التنفيذ المتكررة وكيفية ومجالات إستخدامها وتبين لنا أن الحاسب عندما يبدأ في تنفيذ أي حلقة تنفيذ متكررة تصادفه في البرنامج ، فإنه يتوقف عند تلك الحلقة ويكرر تنفيذ التعليمات التي تحتويها عدداً من المرات يبدأ من ١ إلى ٢ ولا ينتقل الى خارج حلقة التنفيذ الا تحت تأثير أحد إحتالين :

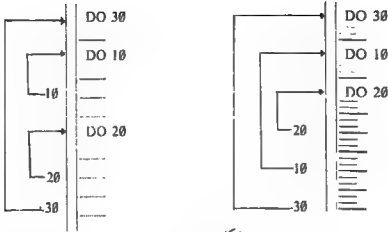
- ١ - أن يصل المتغير I ( الذي يمثل عدداً حلقة التنفيذ ) الى القيمة النهائية ٢ ،
- ٢ - أن تحتوي تعليمات حلقة التنفيذ على إحدى تعليمات التحكم أو الانتقال والتي قد تتسبب في الخروج من تلك الحلقة .

وقد يحتوي البرنامج على اكثر من حلقة تنفيذ كل منها متباعدة عن الأخرى أو إحداها تتبع سابقتها مباشرة كما في شكل (٦-٦) وفي جميع الأحوال فإن الحاسب لن ينتقل الى تنفيذ حلقة منها الا بعد الانتهاء من تنفيذ سابقتها أو الخروج منها.



وحلقات التنفيذ المتكررة المبينة بالشكل (٦-٦) تأتي في شكل متتال إحداها وراء الأخرى ، ولكن هل من الممكن أن تكون هناك اكثر من حلقة تنفيذ متكررة تأتي في شكل متواز ؟ - أي هل من الممكن أن تتخيل وجود الحلقات الثلاث السابقة مثلاً في إحدى الصور التالية في شكل (٦-٧) ؟

شكل (٦-٦)



شكل (٦-٧)

والجواب على ذلك هو أن لغة الفورتران تقبل وجود مثل تلك الحلقات ولكن تحت شروط يجب مراعاتها وستحدث عما فيما بعد ، ولكن ما يجب أن نعرفه الآن أن وجود مجموعة حلقات تنفيذ متكررة كما في الشكلين السابقين يطلق عليها إسم حلقات التنفيذ المتداخلة .

والآن يتبقى السؤال :

كيف يقوم الحاسب بتنفيذ تلك المجموعة من حلقات التنفيذ المتكررة المتداخلة ؟

وللاجابة على ذلك السؤال ، سنحاول أولاً الاجابة على السؤال التالي :

في علم الجبر ، اذا أعطيت الكميّتين الجبريتين التاليتين فكيف يمكنك حلها ؟

$$[A + 3(B - 2C) - 5(3D + 2E)] , [A - 2\{B + 3(C - 2D) + 4\} + 5E]$$

فنحن نعلم من قواعد الجبر البسيطة أنه اذا كانت هناك كميات جبرية كالمية فلحسابها يلزم أولاً فك أو إنهاء الأقواس الداخلية ثم الانتقال بعد ذلك الى الأقواس الخارجية التي تليها وهكذا .

$$[A + 3(B - 2C) - 5(3D + 2E)]$$

فلحساب الكمية الجبرية الأولى :

$$3(B - 2C) = 3B - 6C$$

يجب أولاً حساب القوس الداخلي الأول :

$$5(3D + 2E) = 15D + 10E$$

ثم حساب القوس الداخلي الثاني :

$$A + 3B - 6C - 15D - 10E$$

ثم تنتقل الى فك أو إنهاء القوس الخارجي ليصبح :

$$[A - 2\{B + 3(C - 2D) + 4\} + 5E]$$

ولحساب الكمية الجبرية الثانية :

$$3(C - 2D) = 3C - 6D$$

نبدأ أولاً بفك الأقواس الداخلية من النوع ( ) :

ثم تنتقل الى فك الأقواس الوسطى من النوع { }

$$2\{B + 3C - 6D + 4\} = 2B + 6C - 12D + 8$$

وأخيراً تنتقل الى فك أو إنهاء الأقواس الخارجية من النوع [ ] ، فيكون المقدار هو :

$$A - 2B - 6C + 12D - 8 + 5E$$

وهناك شبه كبير بين طريقة حساب مجموعة من حلقات التنفيذ المتداخلة وطريقة فك مجموعة من الأقواس المتداخلة ، ولذا فلتكن القاعدة العامة الآن هي أنه :

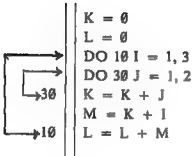
« اذا وجدت مجموعة من حلقات التنفيذ المتداخلة فإن الحاسب يقوم بإنهاء حلقات التنفيذ الداخلية ثم ينتقل الى التي خارجها الى أن يصل الى حلقة التنفيذ الخارجية » .

والفارق الوحيد بين الطريقتين هو أنه بمجرد فك الأقواس فأنتا نستطيع بعد ذلك فك الأقواس التي تليها وهكذا . وفك كل نوع من الأقواس يتم مرة واحدة فقط ، بينما عند حساب مجموعة من حلقات التنفيذ المتداخلة فإنه كلما وصلت حلقة التنفيذ الداخلية الى نهايتها تبدأ حلقة التنفيذ التي خارجها في التغير خطوة واحدة وهكذا حتى تصل الحلقتين التنفيذيتين الى نهايتهما ، عندئذ تبدأ الحلقة التنفيذية التي تليها في التغير خطوة ، وتستمر تلك العملية حتى تصل جميع الحلقات التنفيذية الى قيمة ٠ في كل منها ثم يبدأ الحاسب بعد ذلك في تنفيذ التعليمات التي تلي تلك المجموعة من الحلقات التنفيذية المتداخلة .

وهذه العملية تشبه عداد الكيلومترات في السيارة والذي يتكون من مجموعة متداخلة من التروس ، الداخلي منها يعد الأحاد والذي يتلوه للعشرات ثم المئات وهكذا . وكلما وصل ترس الأحاد الى الرقم 9 يبدأ ترس العشرات في الانتقال خطوة ليبدأ ترس الآحاد مرة أخرى في الدوران حتى الى الرقم 9 ، وهكذا حتى يصل العداد بجميع تروسه الى قيمته العظمى ولكن 99999 .

مثال ٦-٣-٩ :

أوجد قيم  $K$  ،  $L$  من البرنامج التالي :



- عند تنفيذ الحاسب لهذا الجزء من البرنامج فإنه سيضع قيمتي  $K$  ،  $L$  تساوي أصفاراً .
- لفك حلقتي التنفيذ المتكررتين المتداخلتين يبدأ الحاسب بأخذ أول قيمة من المتغير  $I$  . أي يضع  $I = 1$  ثم يني حساب حلقة التنفيذ الداخلية فيأخذ  $J = 1$  ثم ينسب قيمة  $K$  ، ثم يأخذ قيمة  $J = 2$  ليحسب قيمة  $K$

- بوصول قيمة  $J$  الى قيمتها النهائية يبدأ الحاسب في تنفيذ التعليمات التي بعد ذلك حتى يصل الى آخر تعليمة في حلقة التنفيذ الخارجية والتي رقمها 10 .
- بوصول الحاسب الى التعليمة رقم 10 ، يبدأ في تغيير قيمة  $I$  لتكون  $I = 2$  ، ثم يبدأ في فك حلقة التنفيذ الداخلية . أي يأخذ  $I = 1$  ثم  $J = 2$  وهكذا حتى تصل قيمة  $I$  الى 3 وقيمة  $J$  الى 2 . وبالتالي فإن خطوات الحساب ستكون كالتالي :

$$K = 0$$

$$L = 0$$

$$I = 1 :$$

$$J = 1 \rightarrow K = 0 + 1 = 1$$

$$J = 2 \rightarrow K = 1 + 2 = 3$$

$$M = 3 + 1 = 4$$

$$L = 0 + 4 = 4$$

$$I = 2 :$$

$$J = 1 \rightarrow K = 3 + 1 = 4$$

$$J = 2 \rightarrow K = 4 + 2 = 6$$

$$M = 6 + 2 = 8$$

$$L = 4 + 8 = 12$$

$$I = 3 :$$

$$J = 1 \rightarrow K = 6 + 1 = 7$$

$$J = 2 \rightarrow K = 7 + 2 = 9$$

$$M = 9 + 3 = 12$$

$$L = 12 + 12 = 24$$

أي أنه في نهاية تنفيذ حلقتي التنفيذ المتداخلتين ستكون قيمة  $M$  هي 12 ، قيمة  $L$  هي 24 .

مثال ٢-٣-٦ :

في مجموعة الأعداد من 1 الى 9 ، اكتب برنامج لإيجاد مجموع كل ثلاث أعداد متتالية منها ، أي  
يراد إيجاد  $(1+2+3)$  ،  $(4+5+6)$  ،  $(7+8+9)$

```

DO 10 I = 1,7,3
ISUM = 0
I 1 = I
I2 = I + 2
DO 20 J = I1, I2
20 ISUM = ISUM + J
10 WRITE (6,15) ISUM
15 FORMAT (15)

```

٦-٤ ملحوظات على إستخدام حلقة التنفيذ المتكررة : DO

ملحوظة ١ :

١ - كما سبق أن أشرنا أنه يجب :

( أ ) أن تكون جميع المتغيرات الصحيحة  $i_1, i_2, i_3$  إما قيماً صحيحة موجبة أو متغيرات صحيحة سبق إعطاؤها قيماً صحيحة موجبة في البرنامج أو خليط من القيم الصحيحة الموجبة والمتغيرات الصحيحة .

( ب ) أن تكون قيمة  $i_1$  أقل من أو تساوي  $i_2$  ، وإذا حدث أن كانت قيمة  $i_1$  أكبر من أو تساوي قيمة  $i_2$  ، فإن الحاسب سيقوم بتنفيذ حلقة التنفيذ المتكرر مرة واحدة فقط مستخدماً قيمة العداد  $i_1 = i_2$  .

```

DO 3 L = 10,5
=====
3 =====

```

مثال :

ستقوم الحاسب في هذه التعليلة بأخذ قيمة  $L$  تساوي  $10$  ، وينفذ بقية التعليمات التي تحويها حلقة التنفيذ المتكرر مرة واحدة فقط . أي يبدأ بعد ذلك مباشرة في تنفيذ التعليمات التي تلي التعليلة التي رقمها 3 :

ملحوظة ٢ : لايحوز تغيير قيمة المتغير  $i$  داخل التنفيذ المتكرر DO .

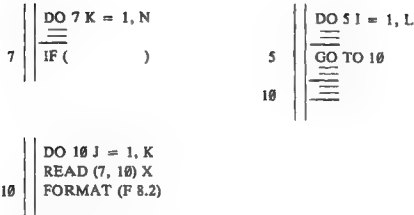
```

DO 10 K = 1,100,2
=====
K = I * 1/2
=====
10 =====

```

مثال :





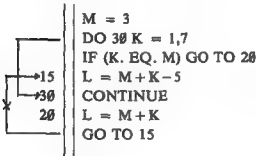
وقد يكون ذلك هو السبب في أن كثيراً من المبرمجين يفضلون إنهاء أي تعليمة حلقة تنفيذ متكرر بتعليمة الاستمرار CONTINUE سواء كان لها أهمية في وضعها أم لا .

ملحوظة • :

يمكن الخروج من حلقة التنفيذ المتكرر ولكن لا يمكن الدخول فيها .



في هذا المثال عندما تصل قيمة K الى 3 فإن الحاسب سيتوقف عن تنفيذ بقية تكرارات الحلقة و ينتقل الى التعليمة التي رقمها 20 والتي توجد خارج نطاق حلقة التنفيذ المتكررة . ولكن لا يمكن كتابة التعليمات كالتالي مثلاً :

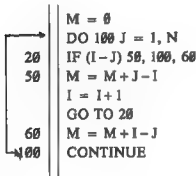




حيث أنه لا يجوز الدخول في حلقة التنفيذ المتكررة عند موضع أي تعليمة منها سواء كان الدخول من تعليمة تسبق حلقة التنفيذ المتكررة أو من تعليمة تأتي بعد حلقة التنفيذ المتكررة كما هو مبين في الجزء السابق من البرنامج .

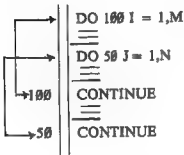
#### ملحوظة ٦ :

داخل حلقة التنفيذ المتكررة يمكن الانتقال من تعليمة الى تعليمة أخرى سابقة لها أو لاحقة . فعلى سبيل المثال فإن الجزء التالي من البرنامج مسموح به ، حيث أن عملية الانتقال تتم داخل حلقة التنفيذ نفسها .

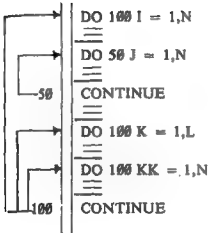


#### ملحوظة ٧ :

في حلقات التنفيذ المتداخلة يجب أن تقع حلقات التنفيذ الداخلية بأكملها داخل حلقات التنفيذ الخارجية ولا يجوز أن تتقاطع حلقتي تنفيذ . فعلى سبيل المثال فإن الجزء التالي من البرنامج غير مسموح به حيث أن هناك تقاطع بين حلقتي تنفيذ متكرر .



كما يجوز أن تشترك أكثر من حلقة تنفيذ متكرر متداخلة في تعليمة نهاية واحدة . فعلى سبيل المثال ، فإن جزء البرنامج التالي مسموح به حيث لا يوجد تقاطع بين حلقات التنفيذ بينما تشترك أكثر من حلقة تنفيذ متكررة في نهاية واحدة .



وفي النهاية بأفترض أن العلامة تمثل حلقة تنفيذ متكررة وأن العلامة تمثل تعليمة انتقال ، فاننا يمكن تلخيص الملاحظات ٥ ، ٦ ، ٧ بالشكال التالية :

ملحوظة رقم	مسموح به	غير مسموح به
٥		
٦		
٧		

شكل (٦-٨)

## ٥-٦ : تمارين على استخدام حلقات التنفيذ المتكرر DO loops

١ - في تعليمات حلقات التنفيذ المتكررة التالية ، أوجد عدد مرات التكرار والقيمة النهائية التي سيأخذها المتغير الصحيح الذي يمثل عدد الحلقة .

- a) DO 50 K = 3,16,4      c) DO 20 KOUNT = 2,9,4  
b) DO 10 L = 2,14,3      d) DO 15 M = 75,82,10

٢ - أوجد الأخطاء - إن وجدت - في التعليمات التالية :

- a) DO 20 L100 = L200, L300, L400  
b) DO 9 K = 2, 7 \* M, 3  
c) DO 10, J = 5,20,K,  
d) DO 5 L = 1,J,L,

٣ - أوجد قيمة L بعد تنفيذ أجزاء البرامج التالية :

a)

```

10 | I=2
   | J=3
   | DO 10 K = 3,14,I
   | L = J+K
   | L = 2*L

```

b)

```

15 | K=5
   | L=0
   | DO 15 I = 1,9,2
   | L = L+K+I

```

c)

```

8 | L=2
  | I=3
  | DO 8 K = 1,5
  | L = L+K+I
  | IF (L. GT. 22) GO TO 5
  | CONTINUE
5 | L = L+5

```

d)

```

15 | L=19
   | M=2
   | DO 15 J = 2,9,M
   | IF (J. GT. 6) GO TO 15
   | L = L-2*I
   | CONTINUE
   | L = M*L

```

c )

```

      L = 19
      M = 2
10    DO 15 J = 2,9,M
      IF (J. EQ. 6) GO TO 10
      L = L - 2*J
15    CONTINUE
      L = M*L

```

٤ - أوجد الأخطاء في أجزاء البرامج التالية :

a )

```

      ==
10    X = A + B
      ==
      DO 50 L = 1,9,2
      ==
15    IF (X.LT.FLOAT (L)) GO TO 50
      ==
      GO TO 10
      ==
50    CONTINUE
      ==
      X = X - B
      ==
      GO TO 15

```

b)

```

      ==
10  X = A + B
      ==
      DO 50 L = 1,9,2
      ==
      IF (X.LT.FLOAT (L)) GO TO 10
      ==
15  X = X - B
      ==
      DO 100 K = 1,7
      ==
      GO TO 15
      ==
50  CONTINUE
100 CONTINUE

```

c)

```

      ==
10  X = A + B
      GO TO 30
15  DO 20 L = 1,7
      ==
      X = X - FLOAT (L)
      ==
50  Y = X*X
      ==
40  POLYN = Y+X-3.
      ==
      DO 20 K = 2,9
      ==
      IF (POLYN. GT 7.) GO TO 40
      ==
30  POLYN = POLYN - 3.
      ==
20  CONTINUE
      ==
      GO TO 50
      ==

```

d)

```

=====
GO TO 30
=====
DO 10 K = 1,7
=====
15 L = K+2
=====
DO 20 M = 1,6
=====
IF (M. EQ. 3) GO TO 30
20 CONTINUE
30 M = M+L
35 DO 50 L = 1,5
=====
GO TO 40
=====
50 CONTINUE
=====
GO TO 35
40 K = K+L
=====
10 CONTINUE
=====

```

٥ - اكتب جزء من برنامج يكتب الرقم 20 عشرون مرة ، الرقم 19 تسعة عشرة مرة وهكذا .

٦ - اكتب جزء من برنامج يكتب الأعداد الفردية الواقعة بين العددين 1 ، N وبجانب كل عدد يكتب مربعة ومكعبة .

٧ - بافترض عددين موجبين N ، K حيث K أكبر من N . اكتب جزء من برنامج باستخدام حلقة التنفيذ المتكرر لكتابة العدد N ثم العدد (N-1) ، (N-2) وهكذا حتى العدد K .

٨ - بافترض أن قيم المتغيرات A ، B ، N معطاه ، اكتب جزء من برنامج لحساب :

$$S = \frac{1}{A} + \frac{1}{A+B} + \frac{1}{A+2B} + \dots + \frac{1}{A+NB}$$

٩ - بافترض أن قيمة المتغير N معروفة ، اكتب برنامج لحساب :

$$\text{Result} = \frac{1}{1^2} \cdot \frac{3}{2^2} \cdot \frac{5}{3^2} \dots \frac{2N-1}{N^2}$$

١٠ - اكتب برنامج يعطي جميع الأعداد الفردية الصحيحة الموجبة والتي أقل من 100 ، دون إعطاء تلك الأعداد الفردية الموجبة التي تقبل القسمة على 7 .

١١- الأعداد الثلاثة الموجبة الصحيحة A, B, C حيث A أقل من B , B أقل من C تكون مثلثا قائم الزاوية اذا كان  $A^2 + B^2 = C^2$  , اكتب برنامج ليجاد جميع قيم C, B, A التي تكون مثلثات قائمة الزاوية حيث A , B أقل من 25 .

١٢- بافراض مجموعة من البطاقات عددها N , على كل منها رقم صحيح موجب . اكتب برنامج ليجاد عدد الأرقام الزوجية وكذلك عدد الأرقام الفردية .

١٣- بافراض المعادلة :  $Y = 2X^3 - 5X^2 + 3X + 2$

اكتب برنامج يحسب قيمة Y التي تحدث على المتغير X عندما :

(أ)  $7 \leq X \leq 7$  - زيادة قيمة X في كل مرة بمقدار 0.1

(ب)  $X \leq 1$  - زيادة قيمة X في كل مرة بمقدار  $\frac{1}{N}$  حيث المتغيرات

I, J, N تكون قد سبق قراءتها وفيها المتغير I أقل من المتغير J .

١٤- بافراض أن هناك ٢٥ رقما صحيحا موجبا يتم ادخالها الى الحاسب عن طريق البطاقات .

اكتب الجزء من البرنامج الذي يعطي ثاني اكبر تلك الأعداد .

١٥- بافراض المعادلة :  $Z = X^2 - 2XY + 3Y^2 - 8X + 3Y - 8$

اكتب برنامج يحسب قيمة Z عند قيم متغيرة للمتغيرين X , Y تتغير من -3 الى 3 وكل منها يزيد بمقدار 0.2 .

١٦- في الهرين السابق ، أوجد اكبر قيمة يأخذها المتغير Z عند قيم Y, X المختلفة .

١٧- يمكن حساب قيمة النسبة التقريبية  $\pi$  باستخدام كثيرة الحدود :

$$\frac{\pi^2}{8} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \dots$$

اكتب برنامج لحساب :

(أ) مجموع المائة حد الأولى من كثيرة الحدود .

(ب) مجموع المائة حد الأولى من كثيرة الحدود بترتيب عكسي .

(ج) أوجد الفرق بين التيجتين .

١٨- مجموعة من الأعداد عددها N يتم ادخال احداها وراء الآخر . اكتب برنامج لحساب اكبر تلك الأعداد وأصغرها .

١٩- بافراض مجموعة من البطاقات على كل منها رقم صحيح موجب . اكتب برنامج ليجاد اكبر عدد زوجي في هذه المجموعة . أو يكتب العنوان التالي :

«NO EVEN INTEGER» في حالة ما اذا كانت جميع الأعداد فردية .

٢٠- بافراض مجموعة من القراءات عددها N تمثل ظاهرة معينة لتغير X . اكتب برنامج يحسب من هذه القراءات الانحراف المعياري لها STDV . استخدم القانون :

$$STDV = \sqrt{\frac{\sum x_i^2}{N} - \left(\frac{\sum x_i}{N}\right)^2}$$

٢١- يتم تحويل الدرجة X بالنظام « الستيني » الى الدرجة XR بالنظام « الدائري » بأستخدام القانون :

$$XR = X \cdot \pi / 180$$

حيث  $\pi$  هي النسبة التقريبية والتي تساوي 3.141592654

- اكتب برنامج لحساب XR لقيم من X تتغير من ٩٠ الى ١٨٠ وكل درجة تزيد عن سابقتها بمقدار نصف درجة .

- استخراج النتائج على الصورة التالية :

#### DEGREES\CONVERSION\ TABLE

X	XR
xxx	xx.xxxxx



## « تمارين »

١ - أوجد الخطأ في كل تعليمة من تعليمات حلقة التنفيذ المتكرر التالية ( إن وجد ) :

- a ) DO 10, I = 1, N
- b ) DO 15, FROM I = 30 TO N-2
- c ) DO 10 I = 1, N, L-1
- d ) DO 20 J = 12, 62, I
- e ) DO 8 MEAN = MABLE, ABLE, NAGLE
- f ) DO 7 I = 3, 1, 3

٢ - في أجزاء البرامج التالية ، أوجد الأخطاء ( إن وجدت ) :

- a )
 

DO 3 I = 1, 6	
DO 7 K = 1, I	
DO 3 J = 1, K	
7    K = K+1	
3    M = I*J*K	
- b )
 

:	
10    L = L+1	
DO 20 L = 1, 5	
15    IF (L. EQ. 6) GO TO 10	
20    CONTINUE	
L = L+1	
GO TO 15	
:	

c)

```

      :
      GO TO 25
      DO 10 I = 1, 3
15    II = II + I
      DO 20 J = 1, 3
      IF (J. EQ. II) GO TO 25
20    CONTINUE
25    JJ = JJ + J
      DO 30 K = 1, 6
      IF (JJ. EQ. K) GO TO 35
30    CONTINUE
35    MULT = I*JJ*K - 10
      IF (MULT. GT. 0) GO TO 25
10    CONTINUE
      GO TO 15

```

٣ - باستخدام تعليمة التنفيذ المتكرر ، اكتب برنامج لحساب :

$$SUM = \frac{1}{1(A+B)} + \frac{1}{2(A+2B)} + \frac{1}{3(A+3B)} + \dots + \frac{1}{N(A+NB)}$$

٤ - اكتب برنامج لحساب الدرجات المئوية  $T_c$  بدلالة الدرجات الفهرنهايتية  $T_F$  باستخدام العلاقة :

$$T_c \equiv \frac{5}{9} (T_F - 32)$$

حيث  $T_F$  تأخذ القيم بين ٣٢ و ٢١٢ بزيادة قدرها  $\frac{1}{4}$

٥ - يمكن حساب النسبة التقريبية  $\pi$  من العلاقة :

$$\pi^2 \approx 6 \left( 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \dots \right)$$

اكتب برنامج لحساب  $\pi$  واحسب عدد الحدود  $N$  التي يجب جمعها بحيث أن :

$$|\pi^2(N) - \pi^2(N-1)| \leq 0.000001$$

٦ - بافتراض أن :

$$R = A^3 + 2AB - 5B^2 + 6A - 8B - 3$$

اكتب برنامج لحساب قيمة  $R$  لقيم مختلفة من  $B, A$  تتغير من -5 إلى +5 بزيادة قدرها

0.1

٧ - اكتب برنامج لحساب مضروب أي عدد صحيح موجب  $N$  ، حيث مضروب  $N$  ( $N!$ ) تعطي من :

$$N! = N \cdot (N-1) \cdot (N-2) \dots 3 \cdot 2 \cdot 1$$

٨ - صيغة ستولنج التقريبية لحساب مضروب أي عدد صحيح موجب  $N$  هي :

$$N! \approx (2\pi N)^{\frac{1}{2}} \left(\frac{N}{e}\right)^N$$

حيث :  $\pi$  هي النسبة التقريبية وتساوي تقريباً 3.141592654  
 $e$  يتم حسابها عن طريق القانون :

$$e \approx 1 + \frac{1}{1!} + \frac{1}{2!} + \dots \approx 2.7182818$$

اكتب برنامج لحساب مضروب أعداد بين 1 و 10

٩ - يمكن حساب جيب أي زاوية مقدرة بالتقدير الدائري من القانون :

$$\sin X = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \dots$$

اكتب برنامج لحساب جيب زوايا بين 0.1 و 5.0 بزيادة قدرها 0.3

١٠ - لحساب الجذر التربيعي لرقم حقيقي موجب  $T$  فإن إحدى الطرق لتحقيق ذلك هي طريقة نيوتن والتي تفترض قيمة تقريبية أولية  $X$  للجذر ، ثم يتم حساب قيمة  $EST$  أقرب إلى الجذر حيث :

$$EST = 0.5 \left( X + \frac{T}{X} \right)$$

ثم يتم ابدال القيمة  $X$  بالقيمة المحسوبة  $EST$  وهكذا حتى يتحقق الشرط :

$$\frac{X - EST}{X} \leq \pm E$$

حيث  $E$  قيمة صغيرة جداً يتوقف إختيارها على درجة التقريب المطلوبة .

١١ - إحدى الطرق لحساب الجذر النوني ( $N$ ) لأي عدد صحيح موجب  $T$  هو إفتراض قيمة تقريبية أولية للجذر ولتكن  $X$  ، ثم حساب قيمة أقرب للجذر  $EST$  بإستخدام القانون :

$$EST = \frac{1}{N} \left( (N-1)X + \frac{T}{X^{N-1}} \right)$$

ثم يتم ابدال القيمة  $X$  بالقيمة المحسوبة  $EST$  وهكذا حتى يكون الفرق بين قيمتين متتاليتين للمتغير  $EST$  أصغر من قيمة صغيرة جداً  $E$  ، يتوقف إختيارها على درجة التقريب المطلوبة .  
 اكتب برنامج لتحقيق ذلك .



## الفصل السابع

### المصفوفات



## الفصل السابع المصفوفات ARRAYS

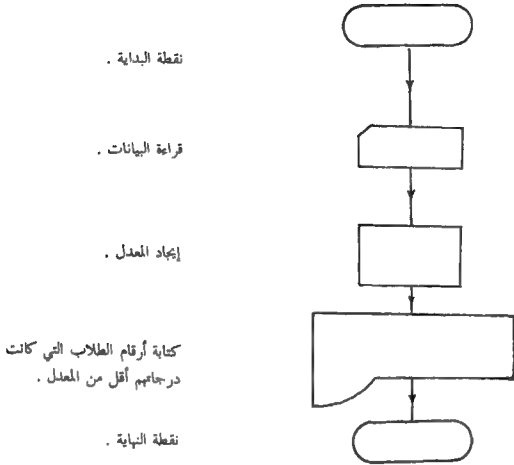
### مقدمة :

تعلمنا الى الآن قدراً جيداً من لغة الفورتران بما يكفي لتنفيذ أي مقدار من العمليات الحسابية المعقدة ، ولكن هناك مجموعة خاصة من المسائل التي تحتاج الى تعليمات جديدة لتنفيذها رغم أن القدر الذي تعلمناه وحتى الآن يمكن استعماله في حل مثل هذه المسائل ولكن ذلك سيؤدي الى كتابة برامج ضخمة وطويلة وعملة مما يؤدي الى فقدان عامل السرعة واختصار الوقت والجهد والمال التي يتميز بها الحاسب الآلي . وبصفة عامة فإن هذه المسائل تتكون غالباً من تطبيقات عملية حيث تحتاج فيها الى استعمال بعض البيانات اكثر من مرة ، فبدلاً من إعطاء كل رقم في البيان اسم خاص به نقوم باعطاء جميع عناصر البيان نفس الاسم وذلك باستخدام مايسمى بالمصفوفات **ARRAYS** .

### ٧-١ : أهمية المصفوفات :

لننظر الى المثال التالي لتدرك أهمية استعمال المصفوفات في لغة الفورتران . لنفرض أن أحد الفصول الدراسية المكون من عشرة طلاب قد قدم أحد الامتحانات العامة في مادة ما ولنفترض أن لدينا بيان يتكون من أرقام الطلاب ودرجاتهم التي حصلوا عليها في هذا الامتحان ، ولنفترض أن البيان مكون من عشرة اسطر بحيث يحتوي كل سطر على رقم الطالب والدرجة التي حصل عليها والآن لنحاول أن نكتب برنامجاً بلغة الفورتران يوجد أرقام الطلاب الذين حصلوا على درجات أقل من المعدل العام في هذا الامتحان .

في البداية نقوم برسم مخطط التدفق للبرنامج .



الشكل (٧-١)

أما نفس البرنامج فسوف يكون كالآتي :

COMMENT ---- PROGRAM TO LIST THE STUDENTS NUMBERS WITH

C

```

BELOW AVERAGE GRADE IN A MATH. TEST.
INTEGER SN1,SN2,SN3, SN4, SN5,SN6,SN7,SN8,SN9,SN10
REAL G1,G2,G3,G4,G5,G6,G7,G8,G9,G10
READ (5,20) SN1, G1
READ (5,20) SN2, G2
READ (5,20) SN3, G3
READ (5,20) SN4, G4
READ (5,20) SN5, G5
  
```



```

      READ (5,20) SN6, G6
      READ (5,20) SN7, G7
      READ (5,20) SN8, G8
      READ (5,20) SN9, G9
      READ (5,20) SN10, G10
20    FORMAT (I6, F5, 1)

C
      AVE = (G1 + G2 + G3 + G4 + G5 + G6 + G7 + G8 + G9 + G10)/10.0

C
      WRITE (6,30)
30    FORMAT ('1 STUDENTS WITH BELOW AVERAGE GRADES'/)

C
      IF (G1. LE. AVE) WRITE (6, 40) SN1
      IF (G2. LE. AVE) WRITE (6, 40) SN2
      IF (G3. LE. AVE) WRITE (6, 40) SN3
      IF (G4. LE. AVE) WRITE (6, 40) SN4
      IF (G5. LE. AVE) WRITE (6, 40) SN5
      IF (G6. LE. AVE) WRITE (6, 40) SN6
      IF (G7. LE. AVE) WRITE (6, 40) SN7
      IF (G8. LE. AVE) WRITE (6, 40) SN8
      IF (G9. LE. AVE) WRITE (6, 40) SN9
      IF (G10. LE. AVE) WRITE (6, 40) SN10
40    FORMAT (' ',16)

C
      STOP
      END

```

data

```

816439 B 72.5
813951 B 85.0
804613 B 69.3
825601 B 90.0
804501 B 58.5
819456 B 77.0
826351 B 80.0
819678 B 62.5
806351 B 74.0
813532 B 66.0

```

output

## STUDENTS WITH BELOW AVERAGE GRADES

816439

804613

804501

819678

813532

الشكل (٧-٢)

في الواقع أنه إذا ماتفحصنا البرنامج تفحصاً دقيقاً فإنه يتضح لنا أن هذه هي أفضل طريقة لكتابة البرنامج على الوجه المطلوب مستعملين في ذلك ماتعلمناه في الفصول السابقة ، ولكن مهلاً فلربما يقول قائل أنه كان بإمكاننا إستعمال الحلقة التكرارية DO loop وبالتالي إختصار عدد كبير من الأسطر ، أي بدلاً من كتابة عشرة أسطر لحملة READ فإنه بإمكاننا إستعمال المتغيرين SN و G لقراءة القيم العشرة الموجودة في البيان على الوجه التالي :

	INTEGER SN
	SUM = 0.0
	DO 25 I = 1, 10
	READ (5, 20) SN, G
	SUM = SUM + G
20	FORMAT (I6, F5.1)
25	CONTINUE
	AVE = SUM/10.0

الشكل (٧-٣)

وفي الواقع فإن هذا ممكن إذا كان المطلوب هو إيجاد المعدل فقط ولكن كيف يمكننا إيجاد أرقام الطلبة الذين حصلوا على درجات أقل من المعدل في حين أن خليتي SN و G تحويان على رقمين فقط هما الموجودين في نهاية البيان أي رقم الطالب العاشر ودرجته في الامتحان بينا الأرقام الأولى قد أزيلت من الذاكرة كما تعلمنا سابقا .

إذا فالبرنامج المكتوب في الشكل (٧-٣) هو أفضل مايمكن كتابته حالياً على ضوء ماتعلمناه حتى نهاية الفصل السابق . وقد يبدو للقارئ أن طول البرنامج مناسب ومعقول وبالتالي فلا داعي لتعلم اداه جديدة لتبسيط البرنامج اكثر من هذا ، ولكن نفترض أن لدينا مائة طالب أو خمسمائة طالب قد تقدموا لهذا الامتحان ، وعندئذ يتحتم علينا اذا ما أردنا إستعمال هذه الطريقة أن نكتب

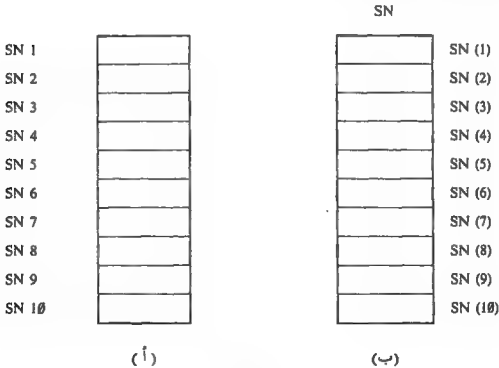
٥٠٠ جملة لقراءة البيان و ٥٠٠ جملة شرطية IF Statement لإنجاد المطلوب وبالتالي فإن المهمة تكون اضاءة للوقت والجهد وموضعاً للسخرية والتهكم .

وللتغلب على هذه المشكلة فأننا نحتاج الى طريقة لتغيير الخلية المستعملة بواسطة جملة أقرأ بدلاً من تغيير الجملة نفسها ، وبالفعل نستطيع أن نفعل هذا بواسطة استعمال المصفوفات ARRAYS .

#### ٧-٢ المصفوفات :

المصفوفة AN ARRAY عبارة عن مجموعة من خلايا الذاكرة التي تحمل نفس الاسم ويمكن التفريق بين عناصر هذه المجموعة بواسطة استعمال دليل عددي index أو subscript يكون مرتبطاً بالاسم المستعمل في البرنامج بحيث يكتب الاسم ثم يكتب على يمينه الدليل العددي بين قوسين كما هو موضح في الشكل (٧-٤) .

وفي المثال السابق الذي استعرضناه مؤخراً نجد أننا نحتاج في الواقع الى مصفوفتين يتكون كل منها من عشرة عناصر ، وعندئذ بدلاً من أن نستعمل عشرة أسماء مختلفة SN1 و SN2 و... و SN10 يمكننا أن نستعمل مصفوفة واحدة تحمل الاسم SN وتتكون من العناصر SN (1) و SN (2) و .. وهكذا الى SN (10)



الشكل (٧-٤)

لاحظ أن ( أ ) يمثل عشرة خلايا تحمل عشرة أسماء مختلفة بينما (ب) يمثل مصفوفة تحمل نفس الاسم SN وتتكون من عشرة خلايا مختلفة تتميز باختلاف الدليل العددي لكل خلية من هذه الخلايا العشر . وفي الواقع لو أن الخاصية الوحيدة لاستعمال المصفوفة هي كتابة (5) SN بدلاً من 5 SN أو (8) SN بدلاً من 8 SN لما استفدنا شيئاً جديداً على الإطلاق ولكن الخاصية الهامة للمصفوفات تكمن في إمكانية كتابة الدليل العددي في صورة مقدار جبري من الممكن أن تتغير قيمته العددية من وقت لآخر خلال تنفيذ البرنامج فمثلاً :

```

      K = 1
      READ (5,500) SN (K), G (K)

```

تعني أن الحاسب سيقراً قيمتي SN (1) و G (1) لأن قيمة K تساوي 1 ولو كانت قيمة K مساوية 4 لقرأ الحاسب قيمتي SN (4) و G (4) . أما لو كانت قيمة K تساوي 1 فإن جملة :

READ (5, 500) SN (4 \* K + 1), G (K + 2)

تعني أن الحاسب سيقراً قيمتي SN (5) و G (3) وهكذا ، ولا بد لنا هنا أن نلفت الانتباه أن الدليل العددي يجب أن يكون عدداً صحيحاً كما يجب أن يكون موجباً في معظم الأنظمة المتبعة .

### ٧-٣ الجملة المهيئة للمصفوفة : ARRAY declarator

لكي نستعمل مصفوفة في برنامج فأننا نحتاج الى توضيح ذلك للحاسب في بداية البرنامج كما فعلنا سابقاً مع المتغيرات الحقيقية والصحيحة . والجملة المهيئة للمصفوفة تقوم بثلاث مهام رئيسية هي :

- ١ - تحدد اسم المصفوفة .
  - ٢ - تحدد طول المصفوفة .
  - ٣ - تحدد نوعية المصفوفة من حيث أنها من نوع الأعداد الصحيحة فقط أو الأعداد الحقيقية فقط
- علماً بأنه لا يمكن الجمع بين نوعين مختلفين من الأرقام في نفس المصفوفة ، أي أنه لا يمكن للعنصر 3 SN أن يكون عدد صحيحاً INTEGER بينما 6 SN يمثل عدداً حقيقياً REAL .

Array declarator

الجملة المبنية للمصفوفة

Type NAME (LEN)

الشكل العام

أما Type فهي إما أن تكون REAL و INTEGER و NAME اسم للمصفوفة لا يتجاوز الست رموز تتكون من أحرف وأرقام وتبدأ بحرف ولا يحتوي الاسم على علامات خاصة .

و LEN عدد صحيح ثابت أكبر من الصفر .

المعنى : اشعار الحاسب بأن NAME هو اسم لمصفوفة تتكون من عدد من خلايا الذاكرة عددها يساوي LEN ويحتاها هو من نوع Type .

أمثلة :

INTEGER A, BX (16) , STUDNT (34)

REAL NADIA (5) , Z3 (14) , XY

هاتان الجملتان لو وجدتا في برنامج واحد فأنها تأمر الحاسب بتشكيل أربعة مصفوفات هي : BX, STUDNT, NADIA, Z3 حسب الأطوال المعطاه بين القوسين وحسب نوع المتغير المبين بكلمتي INTEGER أو REAL فمثلاً STUDNT اسم لمصفوفة طولها ٣٤ أي أنها تتكون من ٣٤ خلية ذات محتوى من نوع الأعداد الصحيحة .

نبيه :

لا بد لنا هنا من الإشارة الى الفارق بين طول المصفوفة LEN وبين المتغير الذي ترتبه LEN في المصفوفة فمثلاً بالنسبة للمثال الذي تكلمنا عنه في بداية هذا الفصل كان لدينا عشرة أرقام للطلاب رمزناها بـ SN ولهذا فأننا نحتاج الجملة المبنية للمصفوفة SN .

INTEGER SN (10)

وكما هو واضح من الشكل (٧-٤ ب) فإن SN (10) يجد ذاتها تعني العنصر العاشر في المصفوفة بينما SN (10) الموجودة في الجملة أعلاه تعني أن طول المصفوفة هو ١٠ أي أنها تتكون من عشر عناصر .

## ٧-٤ عناصر المصفوفة :

عنصر المصفوفة والذي يشكل خلية ذاكرة في المصفوفة يستعمل تماماً كما تستعمل خلايا الذاكرة الأخرى من حيث وجودها في جمل أخرى من جمل الفورتران كالجمل الشرطية والجبرية وإقرأ واكتب ... الخ ، إلا أن هناك إضافة بسيطة بالنسبة لكتابة العنصر نفسه تملخص في لزوم إضافة الدليل العددي الخاص بذلك العنصر لاسم المصفوفة كما بينا سابقاً . وللعلم فإن العناصر تبدأ دائماً من ١ الى الحد الأعلى الذي يمثل طول المصفوفة ولهذا فإن آخر عنصر في المصفوفة له دليل عددي يساوي طول المصفوفة .

أمثلة : هذه بعض الأمثلة التي توضح كيفية استعمال عنصر ما في المصفوفة .

$$A(5) = 47.8 * B + C1$$

$$READ(6, 506) BX(3 * J - 4)$$

$$IF(T(7) + A(3) .LT. T(4) * 3.8) STOP$$

$$Y(M) = Y(M - 1) + SUM$$

$$X(8) = A(8) * Y(3)$$

أما الآن وباستعمال المصفوفات فانه بإمكاننا إعادة كتابة البرنامج السابق لاجتاد أرقام الطلاب الذين حصلوا على درجات أقل من المعدل العام بصورة أكثر بساطة ووضوحاً .

COMMENT --- PROGRAM TO LIST STUDENT'S NUMBERS WITH BELOW

```

C      AVERAGE GRADE IN A MATH. TEST.
C      INTEGER SN (10)
C      REAL G (10), AVE
C      INTEGER I
C
C      ** STORE DATA
C      I = 1
C      10 READ (5,20) SN(I), G(I)
C      20 FORMAT (I6,F5.1)
C      I = I + 1
C      IF (I .LE. 10) GO TO 10
C
C      AVE = (G(1) + G(2) + G(3) + G(4) + G(5) + G(6) + G(7) + G(8) + G(9) + G(10))
C      /10.0

```

```

C      WRITE (6,30)
30     FORMAT ('1', STUDENTS WITH BELOW AVERAGE GRADES'/')
C
      I = 1
35     IF (G(I) .LE. AVE) WRITE (6,40) SN (I)
40     FORMAT (' ', 16)
      I = I + 1
      IF (I .LE. 10) GO TO 35
      STOP
      END

```

الشكل (٧-٥)

وما لاشك فيه أننا إذا أعطينا الحاسب نفس البيان data الموجود في نهاية البرنامج السابق فإننا بالتأكيد سنحصل على نفس النتيجة output الموجودة في الشكل (٧-٢) .

وعلى الطالب أن يلاحظ الفارق الكبير بين البرنامجين من حيث الوضوح والبساطة وتوفير الوقت سواءً في قراءة البرنامج أو كتابته .

#### ٧-٥: جملة تحديد البعد : DIMENSION Statement

تستعمل هذه الجملة لتحديد اسم وطول المصفوفة فقط دون تحديد لنوعية المصفوفة من حيث كونها ذات قيم عشرية أو صحيحة . أما الشكل العام لها فهو :

**DIMENSION NAME (LEN)**

حيث NAME و LEN قد عرفت سابقاً في شرح الجملة المبنية للمصفوفة . وتفتقد هذه الجملة لميزة هامة هي تحديد نوعية المصفوفة منذ النظرة الأولى كما هو الحال مع الجملة المبنية للمصفوفة التي تكلمنا عنها في الصفحات القليلة الماضية .

مثال :

لنفترض أن لدينا مصفوفة اسمها MILE وطولها 20 وذات قيم عشرية وأردنا إستعمالها ضمن أحد البرنامج ، ففي هذه الحالة يمكننا أن نوصل هذه التعليمات الى الحاسب بطريقتين مختلفتين هما كالتالي :

DIMENSION MILE (20)

REAL MILE

REAL MILE (20)

أو

ومن هنا يتضح تفوق الطريقة الثانية على الطريقة الأولى من حيث قصدها ومدلولها المساوي للطريقة الأولى التي إستعملنا فيها سطرين بدلاً من السطر الواحد . لاحظ أنه في الطريقة الأولى لم نضف طول المصفوفة الى الاسم ، أي أنه لا يجوز لنا أن نكتب :

DIMENSION MILE (20)

REAL MILE (20)

لأن هذا يعني بالنسبة للحاسب أن هناك مصفوفتين مختلفتين تحملان نفس الاسم مما يؤدي الى إرسال إشعار بحدوث خطأ في البرنامج .

DIMENSION MILE (20)

مفصال : لو افترض أن كتبنا الجملة

دون أي إضافة أخرى لتحديد نوعية MILE فإن الجملة صحيحة ولكن طبقاً لقاعدة التعمين التلقائي التي تكلمنا عنها في الفصل الثاني فإن MILE مصفوفة طولها 20 وتحمل قيمة صحيحة .



## « تمارين عامة »

التمرين الأول :

تحت أي الشروط يكون العنصر  $X(I)$  والعنصر  $X(J)$  يمثلان نفس العنصر في المصفوفة  $X$  ؟

التمرين الثاني :

ما الذي سيعطيه الحاسب بعد تنفيذ البرنامج التالي ؟

```

10  INTEGER A (10) , I
    A (1) = 0
    A (2) = 1
    I = 3
    A (I) = A (I-1) + A (I-2)
    I = I + 1
    IF (I .LE. 10) GO TO 10
    WRITE (6,20) A(1), A(2), A(3), A(4), A(5)
    WRITE (6,20) A(6), A(7), A(8), A(9), A(10)
20  FORMAT (5I4)
    STOP
    END

```

التمرين الثالث :

لنفترض أن  $B$  عبارة عن مصفوفة من عشر عناصر وأن  $I$  و  $J$  أسماء متغيرات صحيحة وأن قيمة  $I$  تساوي 3 بينما قيمة  $J$  تساوي 7 . أي من الجمل التالية يعتبر صحيحاً ؟ وإذا لم تكن جملة ما صحيحة .. وضح لماذا ؟

```

B(3) = B(1)
B(I) = B(I-1)
B(J) = B(2 * I-1)
B(4) = B(J+1) * B(I * J - 21)
B(2*I) = B(J+5)
B(1.7) = 0.0

```

### القرين الرابع :

أي من جمل التعيين التالية صحيح ؟

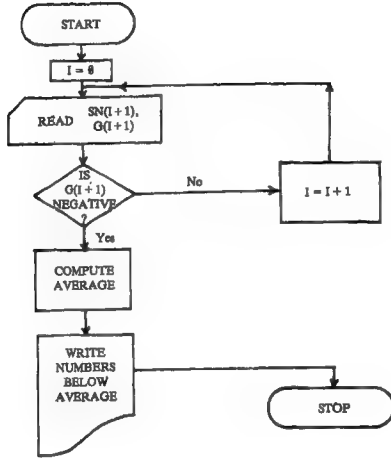
```
REAL A(10)
INTEGER A(13-2)
INTEGER A(I)
REAL X(150) , YFL (B520)
REAL ALI (15.0)
INTEGER SAUDIA (25) XT (3)
```

### حل أفضل :

لنفترض أن لدينا ثلاثون طالباً في الفصل الدراسي الذي تكلمنا عنه سابقاً ولنفترض أنهم أعطوا امتحاناً في مادة ما ولنفترض أننا نريد أن نكتب برنامجاً لاجتاد أرقام الطلاب الذين حصلوا على درجات أقل من المعدل العام للفصل . في الواقع أن البرنامج الموضح في الشكل (٧-٥) يفي بالفرض تماماً إذا ما استبدلنا الرقم 10 بالرقم 30 ولكن على شرط أن يتقدم جميع طلاب الفصل الى الامتحان لأنه في حالة تغيب أحدهم فإن حساب المعدل يختلف كما أن البرنامج لا يصبح صالحاً لاجتاد المطلوب . لذا كان لابد من وجود طريقة لمعرفة عدد الطلبة الذين تقدموا للامتحان . بحيث تكون هذه الطريقة صالحة لأي عدد من الطلبة أقل أو يساوي العدد الكلي للطلبة وهو ثلاثون . أما أفضل الطرق الحالية لاجتاد العدد المتقدم للامتحان فهي معرفة عدد الأرقام التي تقرأ من البيان المعطى للحاسب والذي يشتمل على أرقام ودرجات الطلاب المشاركين في الامتحان . ولهذا فأنا سنضيف في نهاية البيان رقماً سالباً مثلاً يمثل درجة طالب وعندئذ يدرك الحاسب بواسطة إستعمال جملة IF الشرطية بأن البيان الفعلي قد أنتهى فعلاً وأن هذا الرقم السالب الذي قرئ أخيراً يعتبر إشعاراً للحاسب بانتهاء البيان الفعلي .

الشكل (٧-٦) يمثل مخططاً تدفقياً لبرنامجنا المعدل وهو في الواقع يشبه الى حد كبير مخطط التدفق العملي في الشكل (٧-١) مع الاختلاف الوحيد الممثل في تتبع الحاسب للمعد الفعلي للطلبة المتقدمين للامتحان .

ومما تجدر الإشارة اليه هنا أن برنامجاً كهذا يعتبر من البرامج الأساسية التي تتكرر مثيلاتها في المستقبل لدرجة كبيرة . لذا كان من الجدير بالطالب أن يفهم ويدرك النقاط الرئيسية في برنامج كهذا حتى يسهل عليه الكثير في المستقبل .



شكل (٦-٧)

من الملاحظ في هذا المخطط التدفقي أن قيمة  $I$  النهائية ستكون مساوية للعدد الفعلي للطلبة المشاركين في الامتحان لأنه حالما يقرأ الحاسب قيمة سالبة لعلامة الطالب فإنه ينتج الى الخطوة التالية دون أن يشير من قيمة  $I$  النهائية ، وعلى الطالب أن يتأكد من صحة هذا المنطق بتجربة المخطط يدوياً لأربعة أو خمسة طلاب ليتأكد بنفسه من صحة الخطوات المتبعة في المخطط .

أما الآن فإنه بإمكاننا بسهولة أن نكتب البرنامج بعد أن فهمنا مخطط التدفق للعمليات الحسابية والمنطقية التي سيقوم بها البرنامج لتنفيذ الدور المطلوب منه .

COMMENT — PROGRAM TO LIST THE STUDENT S NUMBERS WITH

```

C      BELOW AVERAGE GRADE IN A MATH TEST.
      INTEGER SN (30),I,K
      REAL G(30) , AVE

C
C  **  STORE DATA
      I = 0
10    READ (5,20) SN(I+1) , G(I+1)
20    FORMAT (I6 , F5.1)
      IF (G(I+1) .LT. 0.0) GO TO 30
      I = I+1
      GO TO 10

C
C  **  COMPUTE THE AVERAGE
30    SUM = 0.0
      AI = I
      K = 1
40    SUM = SUM + G(K)
      K = K+1
      IF (K .LE. I) GO TO 40
      AVE = SUM / AI

C
      WRITE (6,50)
50    FORMAT('1 STUDENTS WITH BELOW AVERAGE GRADES')

C
      K = 1
60    IF (G(K) .LE. AVE) WRITE (6,,0) SN(K)
70    FORMAT (' ', I6)
      K = K+1
      IF (K .LE. I) GO TO 60
      STOP
      END

```

الشكل (٧-٧)

#### ٧-٦ المصفوفات وحلقات التنفيذ : ARRAYS AND DO LOOPS

في الواقع من أهم إستعمالات برامج الفورتران هي إستخدام المصفوفات كجزء من حلقات الـ DO فهما في الواقع مكملين لبعضهما لأن إستعمالهما معاً يدل بوضوح على كفاءة الحاسب الآلي ومقدرته العظيمة على إختزال الوقت وتوفير الجهد سواءً من ناحية كتابة البرامج أو من ناحية الوقت الذي يستغرقه تنفيذ البرنامج غالباً .

أما الآن فسنكتب نفس البرنامج السابق بإستعمال حلقات الـ DO وسلاحظ أن البرنامج أصبح أكثر بساطة ووضوحاً .

COMMENT --- PROGRAM TO LIST THE STUDENT'S NUMBERS WITH

```

C      BELOW AVERAGE GRADE IN A MATH TEST
      INTEGER SN (30) , I , K , N
      REAL G (30) , AVE
      SUM = 0.0
C **   STORE DATA AND FIND SUM OF ACTUAL GRADES
      DO 10 I = 1,30
          READ (5,20) SN (I) , G(I)
          IF (G(I) .LE. 0.0) GO TO 30
          SUM = SUM + G(I)
10     CONTINUE
20     FORMAT (I6, F5.1)
      N = I - 1
C **   COMPUTE THE AVERAGE
30     AN = N
      AVE = SUM / AN
C
      WRITE (6,50)
50     FORMAT ('1 STUDENTS WITH BELOW AVERAGE GRADES'/)
C
      DO 60 K = 1 , N
          IF (G(K) .LE. AVE) WRITE (4,70) SN (K)
60     CONTINUE
70     FORMAT (' ', I6)
      STOP
      END

```

الشكل (٧-٨)

٧-٧ بعض الأخطاء الشائعة عن المصفوفات :

عادة ما تكون المصفوفات مصدر إرباك للمبرمج المبتدئ . ولذلك فأن القائمة التالية ستساعد القارئ على تلافي كثير من الأخطاء الشائعة عن إستعمال المصفوفات

أولاً : ليست هناك علاقة بين قيمة الدليل العددي وبين العنصر نفسه فمثلاً ليست هناك علاقة بين العنصر  $SN(5)$  والرقم 5 الذي يمثل الدليل العددي للعنصر .

ثانياً : عندما يستعمل أحد المتغيرات كدليل عددي فإن الأهمية تكمن في قيمة المتغير وليس في اسم المتغير . فمثلاً قد تستعمل في إحدى مراحل البرنامج المتغير  $A(I)$  بينما حيناً آخر قد تستعمل المتغير  $A(J)$  على الرغم من أننا نتعامل مع نفس المصفوفة . ومن الجائز أن يكون  $A(I)$  و  $A(J)$  يرمزان الى نفس العنصر تماماً اذا كانت قيمتي  $I$  و  $J$  متساويتان في إحدى مراحل البرنامج .

ثالثاً : قد يستعمل نفس المتغير كدليل عددي لمصفوفتين مختلفتين ، فمثلاً يجوز لنا أن نستعمل  $A(I)$  و  $B(I)$  في نفس الوقت كما فعلنا في البرنامج السابق عند استعمال  $SN(I)$  و  $G(I)$  . مرة أخرى قيمة الدليل أهم من اسمه .

رابعاً : لاستعمل المصفوفات اذا كنت في غنى عنها ، لأن ذلك قد يؤدي الى كتابة برامج غير مجدية اقتصادياً .

خامساً : القاعدة العامة في لغة الفورتران أن الحاسب يستطيع أن يتعامل مع عنصر واحد في وقت واحد ، أي أنه لا يستطيع أن يتعامل مع أكثر من عنصر في مصفوفة في وقت واحد - فمثلاً في البرنامج السابق لو وضعنا الجملة التالية :

IF (G .LE. AVE) WRITE (6,30) SN

لا تعني أن الحاسب سوف يختبر جميع عناصر المصفوفة  $G$  وفي الواقع أن هذه الجملة غير صحيحة في لغة الفورتران ولذلك لزم أن نقارن كل عنصر على حده كما فعلنا في البرنامج المكتوب في الشكل (٨-٧) .

سادساً : نذكر أنه لا يمكن تغيير طول المصفوفة أثناء البرنامج ، ولذلك إذا لم تكن متأكداً من طول مصفوفة ما ، فما عليك إلا أن تستعمل أكبر طول يمكن أن تصل اليه تلك المصفوفة كما فعلنا بالنسبة للفصل الذي يمكن أن يحتوي على ٣٠ طالب كحد أعلى انظر الشكل (٨-٧) .

## ٨-٧ المصفوفة ذات الدليلين العدديين : Two-Dimensional Arrays

في بعض الأحيان يكون من المفيد أن يستعمل مصفوفة ذات طولين ولها دليلين عدديين قد يكون لهما نفس الطول أو قد يكونا مختلفين . ومثال ذلك  $A(5,3)$  وهنا نذكر أن  $A$  هو اسم لمصفوفة على شكل جدول به خمسة صفوف  $RAWS$  وثلاثة أعمدة  $COLUMNS$  . انظر الشكل (٨-٧) .

## المصفوفة A

	A (1,1)	A (1,2)	A (1,3)
	A (2,1)	A (2,2)	A (2,3)
عدد الصفوف 5	A (3,1)	A (3,2)	A (3,3)
	A (4,1)	A (4,2)	A (4,3)
	A (5,1)	A (5,2)	A (5,3)

عدد الأعمدة = 3

الشكل (٧-٩)

ومن الملاحظ أن الدليل العددي الأول يمثل عدد الصفوف بينما الدليل العددي الثاني عدد الأعمدة . ومما تجدر الإشارة اليه هنا أن الحاسب يقوم بترتيب الخلايا بشكل عمودي كما هو موضح في الشكل (٧-١٠) .

العمود الثاني	A (1,1)
	A (2,1)
	A (3,1)
	A (4,1)
	A (5,1)
	A (1,2)
	⋮
	A (5,2)
	A (1,3)
	⋮
A (5,3)	

الشكل (٧-١٠)

أي أن العنصر الذي ترتيبه ٧ مثلاً هو العنصر  $A(2,2)$  بينما  $A(4,3)$  يحمل الترتيب رقم ١٤ ، والعنصر  $A(5,3)$  ترتيبه ١٥ . أما عدد العناصر التي تتكون منها المصفوفة فهي تساوي حاصل ضرب عدد الصفوف في عدد الأعمدة ، فمثلاً المصفوفة  $A(5,3)$  تتكون من ١٥ عنصراً . ولا يفوتنا هنا أن نؤكد مرة أخرى على أن العنصر الأخير في المصفوفة له دليان عدديان مساويان في الواقع للرقمين الذين يمثلان طول المصفوفة بينما ليست بين الطرفين أي علاقة على الإطلاق .

أمثلة :

REAL X(3,2) , Y(4) , Z	- ١
INTEGER NUMBER (4,2) , C(10)	- ٢
$X(2,2) = X(1,2) * Y(3) + 9.0$	- ٣
$C(5) = A(3) * C(4) / Z$	- ٤
WRITE (6,46) X(1,1) , X(2,1) , X(1,2)	- ٥
READ (5,24) NUMBER (2,1) , C(8)	- ٦

وفي الواقع فأن هناك كثير من التطبيقات العملية التي يظهر فيها مدى الحاجة الى إستعمال هذا النوع من المصفوفات . وهنا نترك لأستاذ المادة مهمة إختيار المثال المناسب لطلبة الفصل الذين يدرسون المادة معه .

#### ٧-٩ ادخال واخراج بيانات المصفوفات :

في هذه المرحلة من الكتاب يعتبر القارئ ملماً بقدر كاف من المعرفة بطرق كتابة المصفوفات ، ومن الملاحظ أنه حتى الآن عند استعمال جملة اكتب WRITE فأن المبرمج عليه أن يعرف تماماً عند كتابة البرنامج عدد عناصر المصفوفة التي يرغب في كتابتها بواسطة الحاسب . وبما لاشك فيه أن القدر الحالي من المعلومات يمكننا من كتابة عناصر أي مصفوفة باستعمال الحلقات التكرارية DO Statements ولكن هذا غالباً ما يؤدي الى كتابة عنصر واحد في كل سطر من البيانات المتحصل عليها . أما اذا رغبتنا في أن نكتب جميع عناصر المصفوفة دفعة واحدة في نفس السطر أو في اكثر من سطر اذا لم يتسع سطر واحد لجميع العناصر ، فأن الأمر يحتاج الى جهد أكبر .

ولتسهيل الأمر فأن هناك طريقة أخرى من طرق ادخال واخراج البيانات تستعمل بصفة خاصة في حالات كهذه مرتبطة بكتابة قيم المصفوفات تسمى الحلقات المباشرة Implied Do List .

وتتلخص هذه الطريقة في كتابة قائمة أسماء المتغيرات ( التي غالباً ماتكون عناصر مصفوفة ) متبوعة بالأدلة العددية التي تحدد عدد مرات تكرار كتابة أسماء المتغيرات في القائمة .



مثال :

نفترض أننا نريد أن نكتب قيم عناصر المصفوفة A من A (1) الى A (N) حيث N اسم لمتغير ذو قيمة عددية صحيحة INTEGER ، فبدلاً من كتابة الحلقة التكرارية .

```

30  I = 1
    WRITE (5,40) A (I)
    I = I + 1
    IF (I .LE. N) GO TO 30

```

نستطيع أن نستغني عن هذا كله بالجملة :

```

WRITE (5,40) (A(I) , I = 1,N)

```

ومن الواضح أن من أهم مزايا استعمال جملة كهذه أنها أبسط كتابة وأسهل قراءة ، ومع ذلك فإن لها ميزة هامة أخرى وهي أنها توجه الحاسب الى كتابة جميع قيم المصفوفة في سطر واحد ( ان اتسم السطر لذلك ) وعند عدم اتساع السطر لجميع القيم فإنه يقوم تلقائياً بكتابة بقية القيم في السطر التالي أو السطور التالية حسب حجم المصفوفة . أما في حالة استعمال الحلقات التنفيذية فإن الأمر يختلف حيث أن الحاسب يقوم بكتابة كل قيمة في سطر لأنه قام الحاسب بتنفيذ جملة اكتب فإنه يبدأ بسطر جديد .

وفي الواقع فإن الحلقة المباشرة Implied Do List لا يشترط فيها أن تكون مشابهة تماماً للجملة التي كتبناها في الفقرة الماضية ، فقاومة المتغيرات يمكن أن تحتوي على أكثر من متغير ، كما أن الدليل العددي يمكن أن يبدأ بأي عدد صحيح موجب أكبر من الواحد كما لا يشترط أن تكون الزيادة بمقدار 1 في كل مرة . فإذا يمكن للدليل العددي أن يبدأ بأي عدد صحيح موجب positive INTEGER كما يمكن أن يزيد كل مرة بأي عدد صحيح موجب . كما يمكن لنا أن نستعمل أسماء متغيرات لتقوم مقام الثوابت التي تحدد بداية الدليل العددي ونهايته والزيادة في كل مرة .

مثال :

الجملة التالية تعتبر جملة صحيحة :

```
WRITE (5,270) (L , A (L) , B (L) , L = M, N, K)
```

فهي تأمر الحاسب بأن يعطي L قيمة متكافئة تساوي M ثم يكتب قيمة كل من L , A (L) و B (L) ثم يزيد قيمة L بمقدار K ثم يكتب قيم L , A (L) , B (L) . وهكذا يستمر التكرار طالما أن قيمة L بعد كل زيادة لم تتجاوز قيمة N المعروفة سلفاً لدى الحاسب .

## « تمرين »

افرض في المثال السابق أن قيمة M تساوي 2 وقيمة N تساوي 10 وقيمة K تساوي 3 ، اكتب جزءاً من برنامج يقوم بنفس مهمة الجملة أعلاه بدون استعمال الحلقات المباشرة ؟  
إذا يمكن تلخيص الشكل العام للحلقة المباشرة كالتالي :-

Implied Do List

الحلقة المباشرة

READ / WRITE (List , V = s, b,i)

الشكل العام

حيث أن List عبارة عن أي قائمة صحيحة .

V عبارة عن اسم الدليل العددي المتغير و s, b, i أعداد صحيحة ثابتة

INTEGER Constants أو اسماء متغيرات صحيحة ليست عنصراً في مصفوفة وذات

قيم موجبة .

المعنى :

يوجه الحاسب الى قراءة أو كتابة قائمة من المتغيرات المتكررة لكل قيمة من V بدءاً

بقية s وتزداد بمقدار i طالما أن V لا تتجاوز قيمة b .

ملاحظة : عندما تكون قيمة i مساوية 1 فانه بالامكان تجاهلها وعدم كتابتها .

أمثلة :

READ (5,1000) (XT(I) , I= 1,15,3)

WRITE (6,48) (A(K - 2) , K= 3,N)

WRITE (6,600) (NUM, V(NUM) , FX , NUM= L,20,INC)

Array Transmission

انفاذ المصفوفات :

المقصود هنا أنه بمجرد كتابة اسم المصفوفة فقط في قائمة جملة اقرأ / اكتب فإن كامل المصفوفة يمكن ارسالها الى الطابع لكتابتها أو يمكن تلقيها بواسطة الجهاز القارئ لقراءتها .

مثال :

```
REAL A, B, X(3)
READ (5,301) A,X,B
```

هاتين الجملتين صحيحتان ومساويتان تماماً للجملتين التاليتين :

```
REAL A, B, X(3)
READ (5,301) A, X(1), X(2), X(3), B
```

٧-٩ تداخل الطرق المتكررة المباشرة :

بالامكان كما هو واضح في الشكل العام للحلقة المباشرة أن يسمح بتداخل اكثر من حلقة مباشرة . أي أنه بالامكان اكمال حلقة مباشرة داخلية تماماً عند كل تكرار للحلقة المباشرة الخارجية . وللتوضيح فاننا نضع الأمثلة التالية التي سوف توضح كثيراً مما نعني :

مثال (١) :

```
WRITE (6,100) C, ((B(J), (A(I), I = 1,3), J = 1,2),D
```

الجملته :

تعتبر مساوية تماماً للجملته :

```
WRITE (6,100) C, B(1), A(1), A(2), A(3), B(2), A(1), A(2), A(3),D
```

مثال (٢) :

```
WRITE (6,200) ((A(I,J), J = 1,3), I = 1,2)
```

الجملته :

تعتبر مساوية تماماً للجملته :

```
WRITE (6,200) A(1,1), A(1,2), A(1,3), A(2,1), A(2,2), A(2,3)
```

وعموماً فإنه يمكن ادخال أو طباعة عناصر أية مصفوفة بطرق عديدة تختلف فيما بينها تبعاً لعاملين :

- ١ - طرق تنفيذ تعليمات القراءة أو الطباعة .
- ٢ - الجملة الشارحة المرافقة لتلك التعليمات .

وفما يلي سنستعرض بعض الأمثلة ، بعرض طرق ادخال أو طباعة عناصر مصفوفات ذات بعد واحد ( أي ذات دليل عددي واحد ) وذات بعدين Two dimensional array . وما ينطبق على تعليمات القراءة أو الادخال ، ينطبق أيضاً على تعليمات الطباعة .

### أولاً - في حالة المصفوفات ذات البعد الواحد :

في هذه الحالة فإن عناصر المصفوفة المراد ادخالها الى الحاسب أو طباعتها تكون مرتبة في شكل صف أو عمود . وشكل المصفوفة في هذه الحالة غير ذي أهمية ولكن الأهم هو وضع كل عنصر في مكانه المخصص له في المصفوفة ، والذي يتحكم في ذلك هو قيمة الدليل العددي كما تبين من الأمثلة السابقة .

#### مثال (١) :

لنفترض أن هناك ٥٠٠ قيمة يراد قراءتها وتخزينها في مصفوفة ذات بعد واحد X . هناك طريقتين لادخال تلك القيم في الحاسب :

١ - اما بادخال تلك القيم قيمة بعد الأخرى ( أي قيمة في كل سطر ) ، وبم ذلك بطريقتين :

C	METHOD A DIMENSION X (500) DO 8 K = 1, 500 8 READ (5,9) X (K) 9 FORMAT (10 F 8.2)
---	---

وفي هذه الحالة لابد من إعطاء كل قيمة على حده ، حتى مع وجود الجملة الشارحة التي تسمح بأخذ عشرة قيم متتالية في كل مرة ، حيث أن حلقة التنفيذ ستجبر الحاسب في أن يقرأ قيمة واحدة فقط كل مرة . ولذا فعلينا أن ندخل الخمسمائة قيمة في خمسمائة سطر متتالية .

C	METHOD B DIMENSION X (500) READ (5,11) (X(K), K = 1,500) 11 FORMAT (F8.2)
---	--

تعتبر هذه الحالة عكسية للحالة السابقة ، فبينما الجملة الشارحة في الطريقة السابقة تسمح باعطاء الحاسب عشرة قيم في كل مرة ولكن تعليمة القراءة هي التي حددت قيمة واحدة وليس أكثر ، فإنه في هذه الحالة فإن تعليمة القراءة تسمح باعطاء الحاسب أكثر من قيمة في كل مرة ، بينما الجملة الشارحة لتلك التعليمة ستجبر الحاسب في أن يقرأ قيمة واحدة فقط .

٢ - أو بادخال تلك القيم في مجموعات ( أي كل مجموعة من القيم في سطر واحد ) ، وبم ذلك بطريقتين واحدة فقط كالتالي :

9 | DIMENSION X (500)  
 | READ (5,9) (X(K), K = 1,500)  
 | FORMAT (10F8.2)

في هذه الحالة فإن تعليمة القراءة تسمح للحاسب بقراءة قيم المصفوفة كلها في سطر واحد ( إن أمكن ذلك ) ، ولكن الجملة الشارحة مستجير الحاسب في هذه الحالة في أن يقرأ عشرة قيم تتبعها العشرة قيم التالية وهكذا . ولذا فيجب علينا في هذه الحالة أن نعطي الحاسب كل عشرة قيم فقط في سطر حيث أنه لن يشعر الحاسب بالقيم التي تعطى له بعد القيمة العاشرة في نفس السطر ، كما أنه في حالة ما إذا كان عدد القيم في السطر أقل من عشرة ، فإنه سيعتبر أن بقية القيم تساوي صفراً .

ومن الحالات المختلفة السابقة تتضح العلاقة التي يجب أخذها في الاعتبار بين تعليمة القراءة والجملة الشارحة لتلك التعليمة .

#### ملحوظة هامة :

في تعليمات القراءة أو الإدخال يجب ألا يزيد عدد الحروف والأرقام التي يراد ادخالها عن ٨٠ في كل سطر ، بينما في تعليمات الطباعة فإن عدد الحروف والأرقام التي يمكن طباعتها أو اخراجها يتوقف على وحدة الاخراج أو الطباعة ففي الشاشات أو البطاقات لا يجب أن يزيد عن ٨٠ حرفاً ورقماً ، بينما في الطابعات يمكن أن يصل إلى ١٢٠ أو ١٣٢ . ولذا - في المثال السابق - نظراً لأن كل قيمة تأخذ الصورة F8.2 ، كان لزاماً علينا ألا يزيد عدد القيم التي يمكن ادخالها إلى الحاسب عن عشرة قيم في كل سطر . وبناءً عليه فلو أن كل قيمة كانت في الصورة F5.2 مثلاً ، لأمكن أن نعطي الحاسب كل ١٦ قيمة في سطر واحد .

#### ثانياً - في حالة المصفوفات ذات البعدين :

في هذه الحالة فإن كل قيمة يراد ادخالها في المصفوفة أو استخراجها منها ، يتحدد موضعها عن طريق الدلييلين العدديين الذي يحدد أولهما رقم الصف الذي سيم ادخال القيمة فيه وثانيهما يحدد رقم العمود . ولقراءة أو طباعة مصفوفة ذات بعدين ، فإن هناك أكثر من طريقة سنحاول التفرقة بينها بافتراض المثال التالي :

#### مثال (٧) :

لنفترض أن هناك ٥٠٠ قيمة ، يراد قراءتها وتخزينها في مصفوفة X تحتوي على ١٠ صفوف ، ٥٠ عموداً فإن ذلك يمكن تنفيذه بإحدى الطرق - التالية :

طريقة أولى : باعطاء الحاسب كل قيمة في سطر عن طريق الصفوف كالتالي :

```

1  DIMENSION X (10,50)
   DO 2 I = 1,10
   DO 2 J = 1,50
2  READ (5,3) X (I,J)
3  FORMAT (20F4.2)

```

في هذه الطريقة نلاحظ أن :

- ١ - تبعاً لخاصية حلقات التنفيذ المتداخلة ، سنجد أن الدليل J ( رقم العمود ) سيأخذ القيم من ١ الى ٥٠ عند كل قيمة ثابتة من الدليل العددي I ( رقم الصف ) .
- ٢ - مع أن الجملة الشارحة تسمح بقراءة ٢٠ قيمة في كل سطر ، إلا أن تعليمة القراءة ستجبر الحاسب أن يقرأ قيمة واحدة يتحدد موقعها بقيمتي الدليلين J, I .

طريقة ثانية : باعطاء الحاسب كل قيمة في سطر عن طريق الأعمدة كالتالي :

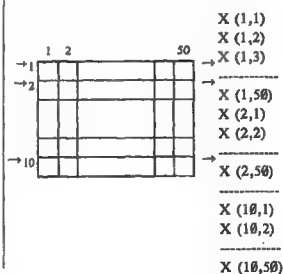
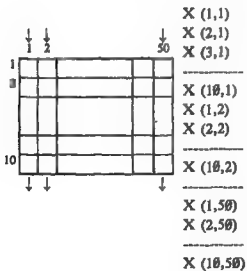
```

1  DIMENSION X (10,50)
   DO 2 J = 1,50
   DO 2 I = 1,10
2  READ (5,3) X (I,J)
3  FORMAT (20 F 4.2)

```

وفي هذه الطريقة قد يتبين الفارق بسهولة عن الطريقة السابقة حيث أن كل ما تم عمله هو تبديل حلقتي التنفيذ بحيث يجب إعطاء الحاسب القيم التي في العمود الأول قيمة بعد أخرى يتلوها قيم العمود الثاني قيمة بعد أخرى وهكذا حتى العمود رقم ٥٠ .

ولذا ففي الطريقتين السابقتين يجب إعطاء القيم كالتالي :



شكل (٧-١١)

طريقة الثالثة : باعطاء الحاسب أكثر من قيمة في كل سطر عن طريق الصفوف كالتالي :

```

1  DIMENSION X (10,50)
2  DO 2 I = 10
3  READ (5,3) (X(I,J), J=1,50)
   FORMAT (20 F4.2)

```

وتحدر هذه الطريقة الأكثر شيوعاً وانتشاراً في التعامل مع المصفوفات ذات البعدين ، حيث يتم فيها كتابة قيم كل صف في المصفوفة في سطر واحد أو مجموعة من السطور . ففي مثالنا هذا يجب إعطاء قيم كل صف في المصفوفة على ثلاثة أسطر بحيث يحتوي كل من السطرين الأولين على ٢٠ قيمة والسطر الثالث على ١٠ قيم فقط . أي أنه لكل قيمة جديدة يأخذها الدليل العددي I ( والذي يمثل رقم الصف ) يجب قراءة ٥٠ قيمة ، حيث أن الدليل العددي J ( والذي يمثل رقم العمود ) يتغير من ١ إلى ٥٠ .

أي أن قيم المصفوفة في تلك الطريقة تعطي كالتالي :

عناصر الصف الأول :

X (1,1)	X (1,2)	.....	X (1,20)
X (1,21)	X (1,22)	.....	X (1,40)
X (1,41)	X (1,42)	.....	X (1,50)

عناصر الصف الثاني :

X (2,1)	X (2,2)	.....	X (2,20)
X (2,21)	X (2,22)	.....	X (2,40)
X (2,41)	X (2,42)	.....	X (2,50)

عناصر الصف العاشر :

X (10,1)	X (10,2)	.....	X (10,20)
X (10,21)	X (10,22)	.....	X (10,40)
X (10,41)	X (10,42)	.....	X (10,50)

شكل (٧-١٢)

ويتضح في هذه الطريقة أن القيمة الأولى من كل صف في المصفوفة تكتب في سطر جديد تتلوها بقية قيم هذا الصف .

طريقة رابعة : باعطاء الحاسب أكثر من قيمة في كل سطر عن طريق الأعمدة كالتالي :

```

2 | DIMENSION X (10,50)
  | DO 2 J = 1,50
3 | READ (5,3) (X(I,J) , I=1,10)
  | FORMAT (20 F4.2)

```

ويتبين من تلك الطريقة أنها مشابهة للطريقة الثالثة ، غير أن الحاسب يقوم بقراءة ١٠ قيم تمثل محتويات العمود رقم J الذي يتغير من ١ الى ٥٠ . ولذا فإن قيمة المصفوفة في تلك الطريقة تعطى كالتالي :

عناصر العمود الأول :  
 $X(1,1) \quad X(2,1) \quad X(3,1) \quad \dots \quad X(10,1)$

عناصر العمود الثاني :  
 $X(1,2) \quad X(2,2) \quad X(3,2) \quad \dots \quad X(10,2)$

عناصر العمود الخمسون :  
 $X(1,50) \quad X(2,50) \quad X(3,50) \quad \dots \quad X(10,50)$

ويتضح من هذه الطريقة أن القيمة الأولى من كل عمود تكتب في سطر جديد تتلوها بقية قيم هذا العمود . كما يتضح في مثالنا هذا أننا لن نستطيع إعطاء الحاسب أكثر من ١٠ قيم في كل سطر ، مع أن الجملة الشارحة تسمح لنا باعطائه ٢٠ قيمة ولكن الدليل العددي I في تعليمة القراءة يتغير من ١ الى ١٠ فقط وبعدها يتغير الدليل العددي J والذي يعطي الإشارة للحاسب بأن يبدأ في قراءة قيم عمود جديد وهكذا .

طريقة خامسة : باعطاء الحاسب قيم المصفوفة في مجموعات متساوية في كل سطر ( عن طريق الصفوف ) .

أي أنه في هذه الطريقة سيم إعطاء الحاسب قيم الصف الأول في سطر أو أكثر ، ثم تكمل مباشرة قيم الصف الثاني ثم تتبعها بقيم الصف الثالث وهكذا بحيث يحتوي كل سطر على مجموعة قيم معينة ( ٢٠ قيمة في مثالنا الحالي ) لزيادة ولا تنقص ، لأنه في حالة زيادتها فإن الحاسب لن يشعر بها . كما أنه في حالة نقصها فإن الحاسب سيعتبر أن القيم المتبقية تساوي أصفاراً مما قد يسبب أخطاءً في نتائج الحسابات قد يصعب اكتشافها بسهولة .



ونكتب صيغة القراءة في هذه الطريقة كالتالي :

```

3 | DIMENSION X (10,50)
  | READ (5,3) (X(I,J), J=1, 50), I=1, 10)
  | FORMAT (20 F4.2)

```

ويمكن اعتبار أن تعلية القراءة في هذه الطريقة تحوي على حلقتي تنفيذ متداخلتين ، إحداهما خارجية والدليل العددي لها هو المتغير الصحيح I ( الذي يمثل رقم الصف في المصفوفة ) ، والأخرى داخلية والدليل العددي لها هو المتغير الصحيح J ( والذي يمثل رقم العمود في المصفوفة ) . ويقوم الحاسب في هذه الطريقة بقراءة قيم المصفوفة جميعها بقسمة كل ٢٠ قيمة على سطر . أي أن قيم المصفوفة في تلك الطريقة يتم إعطاؤها كالتالي :

X (1,1)	X (1,2)	.....	X (1,20)
X (1,21)	X (1,22)	.....	X (1,40)
X (1,41)	X (1,42)	.....	X (2,10)
X (2,11)	X (2,12)	.....	X (2,30)
X (2,31)	X (2,32)	.....	X (2,50)
<hr/>			
X (10,31)	X (10,32)	.....	X (10,50)

شكل (٧-١٣)

طريقة سادسة : بإعطاء الحاسب قيم المصفوفة في مجموعات متساوية في كل سطر ( عن طريق الأعمدة ) :

وهي الطريقة المشابهة للطريقة السابقة ولكن بتعديل بسيط في تعلية القراءة ، وذلك بجعل حلقة التنفيذ التي يمثلها الدليل العددي I كحلقة تنفيذ داخلية وحلقة التنفيذ الأخرى التي يمثلها الدليل العددي J كحلقة تنفيذ خارجية ، مع الأخذ في الاعتبار القيم التي يتغير خلالها كلاً من الدليلين . ولذا فإن صيغة القراءة في هذه الطريقة يمكن كتابتها كالتالي :

```

3 | DIMENSION X (10,50)
  | READ (5,3) (X(I,J), I=1, 10), J=1, 50)
  | FORMAT (20 F 4.2)

```

ولذا فإن قيم المصفوفة في تلك الطريقة ستعطي للحاسب بحيث يكتب له قيم كل عمودين في سطر واحد كالتالي :

## عناصر العمودين ٢،١

X (1,1) X (2,1) ..... X (10,1) X (1,2) X (2,2) ... X (10,2)

## عناصر العمودين ٤،٣

X (1,3) X (2,3) ..... X (10,3) X (1,4) X (2,4) ... X (10,4)

X (1,49) X (2,49) ..... X (10,49) X (1,50) X (2,50) ... X (10,50)

وأخيراً كنوع من التدقيق والمقارنة بين الطرق السابقة نجد أنه يجب أن يتم إعطاء جميع قيم المصفوفة X في مثالنا السابق في عدد من الأسطر كالتالي :

الطريقة	عدد الأسطر
الأولى	٥٠٠ ( كل قيمة في سطر عن طريق الصفوف )
الثانية	٥٠٠ ( كل قيمة في سطر عن طريق الأعمدة )
الثالثة	٣٠ ( كل صف في ثلاثة أسطر )
الرابعة	٥٠ ( كل عمود في سطر )
الخامسة	٢٥ ( كل سطر يحتوي على ٢٠ قيمة مأخوذة عن طريق الصفوف )
السادسة	٢٥ ( كل عمودين في سطر يحتوي على ٢٠ قيمة ) .

## مسألة (٣) :

المصفوفة التالية سبق تخزينها في الحاسب ويراد كتابة قيم كل عمود فيها في سطر . اكتب تعليمتي طباعة مختلفتين لتحقيق ذلك .

$$\left\{ \begin{array}{cccc} 3.07 & -2.1 & 6.75 & -14.8 \\ -6.08 & 7.0 & -0.52 & 3.07 \\ 1.4 & -6.01 & 3.45 & 7.15 \end{array} \right\}$$

أولاً : لنفترض أن المصفوفة السابقة سبق تخزينها تحت اسم المتغير X .

ثانياً : بالنظر في قيم المصفوفة نجد أن أكبر عدد فيها يحتوي على رقمين صحيحين بالإضافة الى الاشارة ، كما أن أقصى عدد من الأرقام العشرية يحتوي - عليها أي عنصر هو رقمين عشريين فقط . وبالتالي فإن كل عنصر في المصفوفة سيأخذ الصورة F 6.2 .

ثالثاً : لكتابة قيم كل عمود في المصفوفة في سطر ، فإنه يمكن إتباع إحدى الطريقتين التاليتين :

C	METHOD A DO 7 J=1, 4 WRITE (6,8) (X(I,J), I=1,3) 7 8     FORMAT (12 F 6.2)
---	--

وفي هذه الطريقة مع أن الجملة الشارحة تسمح بكتابة جميع قيم المصفوفة في سطر واحد ، إلا أن حلقتي التنفيذ ذوات الدليلين I, J بهذا الوضع سيوجب الحاسب على تغيير قيم الدليل I عند كل قيمة ثابتة من J ، أي ستمكن الحاسب من كتابة قيم كل عمود في سطر .

C	METHOD B WRITE (6,9) (X(I,J), I=1,3), J=1,4) 9     FORMAT (3 F 4.2)
---	---

في هذه الطريقة ، فإن تعليمة الطباعة تمكن الحاسب من طباعة جميع قيم المصفوفة عمودا وراء الآخر في سطر واحد ولكن الجملة الشارحة لن تمكن الحاسب من طباعة سوى ثلاث قيم في كل سطر .

مثال (٤) :

في المصفوفة السابقة اكتب ثلاث تعليمات مختلفة تمكن الحاسب من قراءة قيم المصفوفة قيمة بعد أخرى عن طريق الصفوف .

بعد ما تقدم فإنه يمكن كتابة تلك التعليمات كالتالي :

C	METHOD 1 DO 3 I=1, 3 DO 3 J=1, 4 3     READ (5,4) X (I,J) 4     FORMAT (F 4.2)	C	METHOD 2 DO 3 I=1, 3 3     READ (5,4) (X(I,J), J=1,4) 4     FORMAT (F 4.2)
C	METHOD 3 READ (5,4) ((X(I,J), J=1,4), I=1,3) 4     FORMAT (F 4.2)		

وفي الطريقة الأولى فسواء كانت الجملة الشارحة في صورتها الحالية أم مكتوبة بصورة أخرى (28 F 4.2 مثلاً) ، فإن الحاسب لن يقرأ أكثر من قيمة في كل سطر . أما في الطريقتين الثانية والثالثة فلا بد أن تكون الجملة الشارحة بوضعها الحالي ( أي قيمة واحدة في كل سطر ) لأن تعليمات القراءة في هاتين الطريقتين تسمحان بقراءة أكثر من قيمة في كل سطر ولن يمنع الحاسب من تنفيذ ذلك سوى الجملة الشارحة في هذه الحالة .

## « تمارين عامة »

- ١ - ماهو الفرق بين المصفوفة والمتغير العادي من حيث المضمون والشكل العام ؟
- ٢ - كيف يمكننا اشعار الحاسب بأننا نتعامل مع مصفوفة باسم  $X$  طولها 22 ؟
- ٣ - هل يجوز لنا إستعمال مصفوفة طولها 1 فقط ؟
- ٤ - ما نوع المقادير الجبرية التي يمكننا إستعمالها كدليل عددي لمصفوفة ؟
- ٥ - هل يجوز لدليل عددي أن يكون هو نفسه عنصراً في مصفوفة أخرى ، مثل :  $X(N(J))$  ؟
- ٦ - بين الخطأ في كل جملة من الجمل التالية :

DIMENTION X (50)

A (0) = -93

K (K) = NO

A (B/6.7) = 12.0

B (1\*1.) = 13.0

- ٧ - كتب أحد الطلبة البرنامج التالي لقراءة ٥٠ رقماً من ٥٠ سطرأ وحساب معدل هذه الأرقام . أوجد أكبر عدد من الأخطاء في هذا البرنامج :

10	DO 20 K = 1,50
20	READ (5,10) X (K)
	FORMAT (F 10.0)
	CONTINUE
	DO 30 K = 1,50
	SUM = SUM + X (K)
30	CONTINUE
	AVER = SUM/50.0
	WRITE (6,40) AVER
	END

٨ - بدون إستعمال الحلقات المباشرة ، اكتب جمل ادخال أو إخراج بيانات مكافئة للجمل التالية  
على فرضية أن :  $K=2, N=13, M=4$

READ (5,501) (X(J) , J=1,5)

WRITE (6,601) (B(L) , L=M,N,K)

WRITE (6,609) (A(J) , J=K,N,M)

READ (5,20) ((B(I,J) , I=1,M) , J=1,K)

WRITE (6,402) Q,R, (S,B(3,B(3,J),A(J),J=1,K),BC,(A(J),J=1,4)

٩ - إستعمل الحلقات المباشرة لكتابة جمل إدخال وإخراج مكافئة للجمل التالية :

WRITE (6,101) A(1), A(2), A(3), A(4), A(5)

READ (5,706) A(3), A(6), A(9), A(12), A(15)

WRITE (6,292) B(2,1), B(3,1), B(2,2), B(3,2), B(2,3), B(3,3)

READ (5,500) A(2), B(3), A(4), B(4), A(6), B(7)

WRITE (6,260) X(1), Y(4), X(2), Y(7), X(3), Y(10)

١٠- مامي الجمل الغير صحيحة بين جمل الادخال والاخراج التالية :

WRITE (6,2000) (A(J), J = 1,N-1)

READ (5,5000) (J,A(J) , J=1,N)

WRITE (6,6000) (A(J+1) , J=1, C(N))

READ (5,501) (A(I), I = -1,3,2)

WRITE (6,66) A (J)

١١- استعمل جملة DIMENSION واحدة لأشعار الحاسب بأن X مصفوفة طولها 25 و Y مصفوفة طولها 36 بينما Z مصفوفة طولها 15 وجميعها ذات قيم حقيقية .

١٢- لنفرض أن A اسم لمصفوفة تتكون من عشر عناصر . اكتب أجزاء من برامج مختلفة لالبحاز الآتي :

(أ) ضع ناتج حاصل ضرب العنصر الأول في العنصر الثاني في خلية اسمها PROD .

(ب) إستبدل قيمة العنصر الثالث بمعدل العناصر الأولى والثالثة والخامسة .

(ج) إذا كان العنصر الأخير مساوياً للصفر أو موجب ، دعه كما هو ، أما حالة كونه سالباً

فاعكس إشارته .

(د) بإستعمال حلقة DO إستبدل كل عنصر بضعفه .

(هـ) رتب الثلاثة عناصر الأولى ترتيباً تنازلياً .

١٣- المصفوفة B طولها 20 . اكتب أجزاء من برامج مختلفة لتحقيق التالي :

(أ) ضع حاصل قسمة العنصر الرابع على مجموع العنصرين الخامس والسادس في خلية اسمها ABC .

(ب) إستبدل قيم العناصر الأربع الأخيرة بصفر دون إستعمال حلقة DO .

(جـ) اذا كان العنصر العاشر اكبر من 10 ، إستبدله بمعدل العنصرين التاسع والحادي عشر .

(د) إستبدل قيم العناصر الفردية بالقيمة 1- وإستبدل قيم العناصر الزوجية بالقيمة الثانية 1 .

(هـ) رتب الأربعة عناصر الأخيرة ترتيباً تصاعدياً .

١٤- ALI و BADR إسمين لمصفوفتين طول كل منهما 10 . إستبدل قيمة كل عنصر في ALI بقيمة نظيره من BADR والعكس . ضع مجموع عناصر ALI في خلية اسمها SUM وحاصل ضرب عناصر BADR في خلية اسمها PROD .

١٥- A و B مصفوفتين طول كل منهما 15 اكتب برنامجاً يوجد مجموع حاصل ضرب كل عنصر من A في نظيره من B ، ثم أوجد قيمة الجذر التربيعي للنتائج . ( اذا كان موجبا 1 ) .

١٦- X و Y مصفوفتين طول كل منهما 30 . اكتب برنامجاً يقارن كل عنصر في X مع نظيره من Y ويقوم بطبع رسالة خاصة لكل حالة دون طباعة قيمتي X و Y .

مثال : X(3) IS GREATER THAN Y(3)

١٧- اكتب برنامجاً يقوم بمهمة ترتيب عناصر مصفوفة اسمها NUMBER وطولها 1000 ترتيباً تصاعدياً .

ملاحظة : هذا البرنامج يصلح لترتيب الطلبة حسب أرقامهم مثلاً .

١٨- اكتب برنامجاً ينتج جدول ضرب من حجم 12x12 .

١٩- اكتب برنامجاً يقوم بمهمة إيجاد أكبر عنصر وأصغر عنصر في مصفوفة طولها 50 .

٢٠- ادى ٩٦ طالباً امتحاناً في مادة الرياضيات ١٠١ ع . اكتب برنامجاً ينتج جدولاً بسيطاً يوضح توزيع الدرجات كما هو موضح في المثال التالي :

GRADE RANGE	NUMBER OF STUDENTS
98-100	15
88-89	21
78-89	29
68-69	16
58-59	15

٢١- من المقادير الشائعة في العمليات الاحصائية ، المعدل Mean والانحراف القياسي Standard Deviation أما المعدل فيمكن ايجاده باستعمال القانون :

$$\bar{X} = \left( \sum_{i=1}^n x_i \right) / n$$

وأما الانحراف القياسي فيمكن ايجاده بالقانون :

$$s = \left( \sum_{i=1}^n (x_i - \bar{X})^2 / n-1 \right)$$

اكتب برنامجاً لايجاد هذين المقدارين لعينة احصائية تتكون من ٢٠ قراءة .

٢٢- يقوم أحد طلاب مادة الأحياء بدراسة لبعض فئات من الحشرات الصغيرة الموزعة الى عشرين صنفاً تحمل أرقاماً من ١-١٠ . أحد خطوات التجربة تحتاج الى تسجيل وزن الحشرة ونتاج جدول يظهر معدل وزن حشرات كل صنف .

البيان المعطى مكون من عدة مئات من السطور ، يتكون كل سطر منها من قراءتين القراءة الأولى لرقم الصنف في الخانتين الأولى والثانية . أما القراءة العشرية الثانية في الخانات من ٣ الى ٨ وتمثل وزن الحشرة بالجرام ( بما في ذلك الفاصلة العشرية ) .

اكتب برنامجاً يقرأ هذه السطور حتى يجد السطر المزيف حيث رقم الصنف يكون خارج نطاق ( ١ الى ٢٠ ) وهذا بمثابة إشعار بانتهاء البيان المعطى . المطلوب ايجاد عدد الحشرات من كل صنف ومجموع أوزان كل صنف ومن ثم معدل وزن كل صنف .

٢٣- جمع طالب في مادة علم النفس مئات من الدرجات التي استحقها طلاب مختلفون من الجامعة في ثلاثين مادة مختلفة تحمل كل مادة رقم كمبيوتر خاص بها . المطلوب كتابة تقرير يوضح العلامة الكبرى والعلامة الصغرى في كل مادة على حده .

اكتب برنامجاً يتسج بياناً مكوناً من ثلاثين سطرأ يحتوي كل سطر على رقم المادة وعدد الطلاب الذين تقدموا للامتحان فيها بالاضافة الى العلامة الصغرى والكبرى .

٢٤- اكتب برنامجاً يقوم بمهمة الحجز الآتي لخطوط طيران الجامعة التي يتكون أسطولها من سبع طائرات تقوم يومياً بسبع رحلات بساعات مختلفة كما يلي :

رقم الرحلة	السعة راكب
٠٠١	١٥٠
٧٠٧	١٣٢
١٠٦	٢٠٨
٢٠٩	١٣٠
٠٦٢	١٦٦
٢٥٠	١٥٩
٣٠٣	١٧٠

علماً بأن الحجز يتم بطريقة الأول فالأول « بدون واسطة » . المطلوب انتاج بيانين مختلفين يحتوي الأول على رقم الرحلة ثم أسماء المسافرين عليها وأرقام هواتفهم أما البيان الثاني فيحول اسم جميع الأشخاص الحاجزين وأرقام هواتفهم والرحلة التي تم الحجز عليها .



## الفصل الثامن البرامج الجزئية



## الفصل الثامن

### Subprograms

### البرامج الجزئية

#### مقدمة :

تتميز لغة الفورتران عن معظم لغات الحاسبات بإمكانية استخدام ما يسمى « بالبرامج الجزئية » حتى أن بعض لغات الحاسبات الأخرى بدأت في تطوير لغاتها بإضافة هذا النوع من البرامج الجزئية إلى لغاتها الأصلية . وفي الفصل السادس رفعا شعار « كل برنامج يمكن تصغيره أو اختصاره » ، ولعل استخدام البرامج الجزئية يعتبر أقوى سلاح لتحقيق هذا الهدف . ففي أغلب المشاكل العلمية قد يضطر المبرمج إلى تكرار كتابة مجموعة من التعليمات أكثر من مرة في نفس البرنامج ، مما يجعل البرنامج طويلاً ومملأً ويكون عرضه لكثير من الأخطاء . وفي عصرنا الحالي عرفنا طرقاً لحلول مشكلات مختلفة وأطلق على تلك الطرق اسم المحوارزم وعند عمل برنامج لحل إحدى تلك المشكلات فإن المبرمج سيضطر لكتابة مجموعة من التعليمات لتنفيذ الخوارزم المستخدم في حل مشكلته وقد يضطر لتكرار كتابتها أكثر من مرة في نفس البرنامج إن تطلب الأمر ذلك .

فعل سبيل المثال ، لحساب التوافيق  $\binom{N}{R}$  بين عددين  $R, N$  مثلاً فإن الأمر يتطلب حساب مضروب  $N$   $(N!)$  ،  $R!$  ،  $(N-R)!$  حيث أن :

$$\binom{N}{R} = \frac{N!}{R! (N-R)!}$$

وذلك قد يتطلب تكرار كتابة مجموعة التعليمات الخاصة بحساب المضروب ثلاثة مرات في نفس البرنامج . وإذا تطلب الأمر حساب التوافيق أكثر من مرة في نفس البرنامج فقد يستدعي ذلك تكرار كتابة تلك المجموعة من التعليمات عدداً كبيراً من المرات مما يجعل البرنامج طويلاً ومملأً ومكرراً في أكثر أجزائه وعرضه لكثير من الأخطاء . ولكن إذا أمكن وضع مجموعة التعليمات التي تتكرر أكثر من مرة في برنامج جزئي مستقل يتم تخزينه جالباً في وحدة التخزين ويتم استدعاؤه عن طريق البرنامج الرئيسي كلما تطلب الأمر ذلك فإن ذلك لاشك سيوفر الكثير من وحدات التخزين كما سيختصر كثيراً من حجم البرنامج ويجعل التحكم في أجزائه ممكناً وسهلاً . وما ينطبق على مجموعة من التعليمات تتكرر أكثر من مرة في البرنامج ، فإن نفس الشيء قد يحدث لتعليمة واحدة فقط يتم حسابها أكثر من مرة . وفي جميع تلك الحالات فإنه من المستحسن استخدام ما يسمى بالبرامج الجزئية .

وفيما يلي نستعرض الأنواع المختلفة من البرامج الجزئية وكيفية إستخدامها .

## ٨-١ الدوال سابقة التخزين : Library Functions or Preprogrammed Packages

هناك بعض الدوال التي يكثر استخدامها في حل المشاكل الرياضية والهندسية المختلفة مثل إيجاد الجذر التربيعي لقيمة ما أو إيجاد لوغاريتم عدد ما أو إيجاد جيب زاوية أو جيب تمامها ... الخ . مثل تلك الدوال الهامة والتي قد يتطلب الأمر استخدامها أكثر من مرة في البرنامج أو يتطلب استخدامها في أغلب البرامج المختلفة ، يتم تخزينها مسبقاً في الحاسب عن طريق ماسبق أن اطلقنا عليه اسم المجموعات أو المشغلات Compilers . وتتوقف نوعية تلك الدوال وعددها وطريقة استخدامها على نوع الحاسب وطرازه ، ولذا فيجب على كل مبرمج أن يكون على علم بما يحتويه الحاسب الذي سيعمل عليه من دوال مخزنة . وعموماً فإن معظم مشغلات الحاسبات العددية تحتوي على الدوال الآتية :

### ١ - الجذر التربيعي : Square root

عند حساب الجذر التربيعي لأي عدد أو تعبير رياضي فإن ذلك يتم ببساطة وبدون أن يندل المبرمج أي مجهود في كتابة مجموعة التعليمات (\*) التي يتطلبها حساب الجذر التربيعي وذلك عن طريق استدعاء البرنامج الجزئي المكلف بتلك المهمة والذي يكون قد سبق تخزينه في موضع ما من وحدة التخزين للحاسب وذلك باستخدام التعبير  $\text{SQRT}(a)$  حيث  $a$  تمثل ثابت حقيقي أو متغير حقيقي أو تعبير رياضي يحوي مجموعة من المتغيرات الحقيقية .

وبشرط أن  $a$  يجب أن تكون موجبة .

مسألة (١) :  $Y = \text{SQRT}(X)$

في هذا المثال ، فإن الحاسب سيقوم بحساب الجذر التربيعي للمتغير  $X$  ثم يقوم بتخزين تلك القيمة في المتغير  $Y$  وذلك بإفراض أن قيمة  $X$  غير سالبة .

مسألة (٢) :  $\text{ROOT} = \text{SQRT}(B*B - 4.0*A*C)$

بعد حساب التعبير الرياضي  $(B*B - 4.0*A*C)$  وإيجاد قيمته ، يقوم الحاسب باختبار ما إذا كانت تلك القيمة موجبة أم سالبة ، وفي حالة ما إذا كانت موجبة يتم حساب الجذر التربيعي لتلك القيمة وتخزينه في المتغير  $\text{ROOT}$  ، أما إذا كانت سالبة فغالباً ما يعطي الحاسب إشارة خطأ (Error) ويتوقف عن تكملة الحسابات .

(\*) من المعروف في الحاسبات العددية أن هناك طرقاً رياضية كثيرة لحساب أي دالة ، ويمتاز كل طريقة عن الأخرى في درجة دقتها وسرعة الحساب بها .

## ٢ - القيمة المطلقة : Absolute Value

تقوم هذه الدالة عند استخدامها في البرنامج بعمل ما يدل عليه اسمها تماماً ، أي أنها تقوم بإهمال إشارة قيمة المتغير المطلوب إيجاد قيمته المطلقة . والصورة العامة لهذه الدالة :  $ABS(X)$  ، حيث  $X$  تمثل متغير حقيقي أو تعبير رياضي له قيمة حقيقية ، بينما  $K$  تمثل متغير صحيح أو تعبير رياضي له قيمة صحيحة .

$$ROOT = SQRT(ABS(B*B - 4*A*C)) \quad \text{مثال :}$$

ففي هذا المثال ، فإن القيمة التي ستخزن في المتغير  $ROOT$  يتم حسابها بعد أن يقوم الحاسب بمعرفة القيمة المطلقة للتعبير  $(B*B - 4*A*C)$  وأخذ الجذر التربيعي لتلك القيمة . ويلاحظ في المثال السابق أن لكل دالة مجموعة الأقواس الخاصة بها ، ويقوم الحاسب بفك تلك الأقواس طبقاً للطريقة التي سبق شرحها في الفصل الثاني .

## ٣ - الدالة الأسية (\*) Exponential

كثيراً ما نستخدم في حساباتنا العلمية الدالة الأسية  $e^{(2.71828..)}$  والتي عادة ما تكون في الصورة  $e^x$  حيث  $x$  تمثل ثابت حقيقي أو متغير حقيقي أو كمية حقيقية ، ولذا فإن الصورة العامة لتلك الدالة في لغة الفورتران هي  $EXP(x)$  .

$$Y = A * EXP(2.0*A - B**C) \quad \text{مثال :}$$

في هذا المثال يتم حساب المقدار  $(2.0*A - B**C)$  أولاً ، ثم يتم حساب الدالة الأسية لهذا المقدار ويضرب في قيمة المتغير  $A$  ، ثم توضع النتيجة النهائية في المتغير  $Y$  .

## ٤ - دالة اللوغاريتم (\*) Logarithms

أحدى الدوال الأخرى التي يكثر استخدامها في حلول المشاكل الرياضية ، وعن طريق بعض الحاسبات يمكن حساب نوعين مختلفين من تلك الدالة :

١ - دالة اللوغاريتم الطبيعي (أي للأساس  $e$ ) وصورتها العامة في لغة الفورتران هي

$$e^x \text{ يمكن حساب الدالة الأسية } e^x \text{ عن طريق حساب مفكوك المتسلسلة :}$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!} + \dots$$

(\*) إحدى الصيغ الرياضية لدالة اللوغاريتم هي :

$$\text{Log}(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{4} - \dots$$

**ALOG (x)**

٢ - دالة اللوغاريتم العام (أي للأساس ١٠) وصورتها العامة هي **ALOG 10 (x)** وفي كلتا الدالتين فإن x تمثل ثابت حقيقي أو متغير حقيقي أو تعبير رياضي يأخذ قيمة حقيقية أكبر من الصفر .

مثال :

$$R = \text{ALOG } (A^{**}2 + B^{**}2)$$

$$S = \text{ALOG } 10 (z)$$

• - دوال الجيب وجيب التمام والظل : **Sine, Cosine and Tangent**

بعض الحاسبات تحوي تلك الدوال الثلاث ، بينما معظمها لا يحوي غير دالتي الجيب وجيب التمام . وعموماً فإن الصورة العامة لتلك الدوال هي : **SIN (x), COS (x), TAN (x)**

وفي جميع تلك الدوال فإن الزاوية x يجب أن يعبر عنها بالتقدير الدائري ، وإذا لم تكن كذلك فيجب تحويلها أولاً قبل استخدام أي من تلك الدوال . وإذا لم يكن الحاسب المستخدم يحوي دالة الظل فإنه يمكن حسابها بسهولة باستخدام القانون :  $\tan (x) = \sin (x) / \cos (x)$  ، وبشرط أن x لا تساوي :  $(\pi + \frac{\pi}{2})$

حيث n عدد صحيح موجب أكبر من أو يساوي الصفر حيث أن  $\cos (x)$  عند تلك الحالات يساوي صفراً .

مثال :

$$Y = \text{SIN (ANGLE)}$$

$$YD = \text{SIN (XRAD)* COS (YRAD)}$$

$$YAL = Y * \text{TAN (XRAD)}$$

وهناك دوال أخرى بعضها يعتمد على متغير واحد وبعضها الآخر على أكثر من متغير ، وفي الجدول التالي سنذكر بعض أهم الدوال الأخرى التي تكثر استخدامها والتي يحتوي عليها الحاسب  
I.B.M. / 370

استخدام الدالة	المعنى الرياضي	اسم الدالة	عدد المتغيرات	نوع	
				المتغير	النتيجة
تحويل	التحويل إلى قيمة صحيحة	INT IFIX	١ ١	حقيقي حقيقي	صحيح صحيح
	التحويل إلى قيمة حقيقية	REAL FLOAT	١ ١	صحيح صحيح	حقيقي حقيقي
حذف المتبقي	الجزء الصحيح من قيمة متغير	AINT	١	حقيقي	حقيقي
المتبقي	$x - \text{int}(x/y) * y$	AMOD	٢	حقيقي	حقيقي
أكبر قيمة	$\max(x_1, x_2, \dots)$	AMAX 8 AMAX 1 MAX 1	أكبر من أو يساوي ٢	صحيح حقيقي حقيقي صحيح	حقيقي حقيقي حقيقي صحيح
أقل قيمة	$\min(x_1, x_2, \dots)$	AMIN 8 AMIN 1 MIN 1	أكبر من أو يساوي ٢	صحيح حقيقي حقيقي صحيح	حقيقي حقيقي حقيقي صحيح
حساب قيمة زاوية	$\sin^{-1}(x)$	ARSIN	١	حقيقي	حقيقي
	$\cos^{-1}(x)$	ARCOS	١	حقيقي	حقيقي
	$\tan^{-1}(x)$	ATAN	١	حقيقي	حقيقي
	$\tan^{-1}(x/y)$	ATAN 2	٢	حقيقي	حقيقي
دوال زائدية	$\sinh(x)$	SINH	١	حقيقي	حقيقي
	$\cosh(x)$	COSH	١	حقيقي	حقيقي
	$\tanh(x)$	TANH	١	حقيقي	حقيقي

أمثلة على استخدام بعض الدوال السابقة :

$$SINA = \sin(A)$$

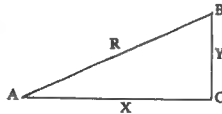
$$B = \arccos(0.5)$$

$$A = \arcsin(\sin(A))$$

$$B = \arctan(X/Y)$$

$$XMIN = \min(X1, X2, X3, X4)$$

$$R = \text{amod}(48.5, 6.3)$$



في التعليم الأخيرة فإن قيمة R تمثل الفارق بين 48.5 وأكبر قيمة صحيحة للكسر  $\frac{48.5}{6.3}$  مضروبة في 6.3 ، أي أن :

$$R = 48.5 - \text{int} \left( \frac{48.5}{6.3} \right) \times 6.3 = 48.5 - 7 \times 6.3 = 4.4$$

#### ٢-٨ الدوال ذات التعبير الرياضي : Arithmetic Statement Functions

عندما يتكرر حساب تعبير رياضي في برنامج ما ، فإن هذا التعبير يمكن أن يوضع في صورة دالة يعطي لها اسم من واضع البرنامج ، وهذا الاسم يختلف عن أسماء الدوال سابقة التخزين التي ورد ذكرها في ١-٨ ، ويتم حساب قيمة تلك الدالة في البرنامج كلما ورد اسمها وقيم المتغيرات التي تعتمد عليها . كما أنه لا يمكن اعتبار مثل تلك الدوال برامج جزئية لسببين :

١ - أن التعبير الرياضي الذي يمثل تلك الدالة ، يمكن اعتباره إحدى التعليمات الواردة في البرنامج ، مثلها مثل أي تعليمة أخرى ، أي أنها تعتبر جزءاً من البرنامج نفسه .

٢ - أن تلك الدالة تكتب في صورة تعليمة واحدة فقط وليس أكثر ، وبالتالي لا يلزم لها أية تعليمة أخرى مثل تعليمة النهاية END Statement التي تشعر الحاسب بأن البرنامج قد أنتهى عند تلك التعليمة .

ويجب أن توضع التعليمة التي تمثل تلك الدالة في أول البرنامج بحيث تسبق أي تعليمة تنفيذية أخرى فيه مثل تعليمات القراءة أو الكتابة أو الحساب .... الخ . والصورة العامة لمثل هذا النوع من الدوال هي :

$$\text{NAME}(a_1, a_2, \dots, a_n) = \text{arithmetic expression in } a_1, a_2, \dots, a_n$$

حيث NAME : يدل على الاسم المرفق للدالة والذي ينطبق عليه نفس شروط إعطاء اسم لأي متغير ، وقيمة الدالة من حيث أنها حقيقية real أو صحيحة integer يتوقف على اسم الدالة وأول حرف فيه أو من تعريف الدالة نفسها مسبقاً .

$a_1, a_2, \dots, a_n$  : متغيرات سيتم استخدامها في حساب قيمة الدالة ويجب أن تكون موجودة داخل التعبير الرياضي المستخدم في حساب قيمة الدالة . وعند حساب الدالة في أكثر من موضع في البرنامج فإنه ليس من الضروري أن تستخدم نفس تلك المتغيرات ولكن يمكن استخدام أسماء ومتغيرات أخرى ولكن بشرط أن تكون مساوية لعدد المتغيرات الأصلية ، وكل متغير يناظر في نوعه من حيث أنه حقيقي أو صحيح للمتغير الأصلي المستخدم في حساب قيمة تلك الدالة .



مثال :

```

10  ROOT1 (A,B,C)= (-B+SQRT(B**2-4.0*A*C))/(2.0*A)
    ROOT2 (A,B,C)= (-B-SQRT (B**2-4.0*A*C))/(2.0*A)
    READ (3,10) X,Y,Z
    FORMAT (3F 7.2)
    :
    :
    S = (X + Y + Z) / 3.0
    R1 = ROOT1 (S,Y,Z)
    R2 = ROOT2 (S,Y,Z)
    WRITE (6,15) R1, R2
15  FORMAT ('R1 =', F12.3, 5x, 'R2 =', F12.3)
    STOP
    END

```

نلاحظ في المثال السابق :

١ - أننا قمنا بتعريف الدالتين مختلفتين أحدهما تسمى ROOT 1 والأخرى تسمى ROOT 2 ولو أن كلاً منهما يعتمد على المتغيرات A, B, C ، وقد تم تعريف هاتين الدالتين وذلك بوضعهما كأول جملتين في البرنامج ، وقبل أي جملة تنفيذية .

٢ - أنه قد تم استخدام الدالتين في بعض أجزاء البرنامج ولكن بمتغيرات أخرى S, Y, Z سبق معرفة قيمها عن طريق تعليمة القراءة مثلاً ، أو عن طريق حسابها في البرنامج. ولذا فإن قيمة S ستستخدم في الدالتين لتحل محل المتغير A وقيمة Y محل المتغير B وقيمة Z للمتغير C .

٣ - أن التعبير الرياضي للدالة قد تحتوي على دوال سابقة التخزين مثل SQRT في مثالنا هذا .

٤ - حيث أن اسم كل دالة يمثل متغير حقيقي ، لذا فيجب أن نتوقع أن قيمة كل دالة منها حقيقية أيضاً .

### ٨-٣ الدوال في صورة برامج جزئية : Function Subprograms

فيما سبق ذكرنا نوعين من الدوال ، أحدهما الدوال سابقة التخزين والتي يكون قد سبق تحضيرها وتخزينها بواسطة الشركة المنتجة للحاسب والتي لا تتطلب جهداً في استخدامها سوى وضع اسم تلك الدالة بمتغيراتها في صورة تعليمة تنفيذية داخل البرنامج ، والأخرى دوال ذات تعبير رياضي

توضع في أول البرنامج وكل دالة تأخذ صورة تعليمية تنفيذية طرفها الأيسر يشمل اسم الدالة والمتغيرات التي تعتمد على حسابها تلك الدالة ، بينما طرفها الأيمن يشمل الصيغة الرياضية المطلوب حسابها . ويتم حساب تلك الدالة في أي موضع من البرنامج توضع اسم الدالة ومتغيراتها ، أو متغيرات نظيرة لها في صورة تعليمية تنفيذية ، بعد إعطاء قيم للمتغيرات الداخلة في حساب تلك الدالة .

وهناك نوع ثالث من الدوال يحتاج لحسابها كتابة أكثر من تعليمية تنفيذية ، وطالما أن الهدف من استخدام تلك الدوال هو الاضطرار إلى حسابها أكثر من مرة داخل البرنامج ، لذا فإن هذا النوع من الدوال يكتب في صورة برنامج جزئي مستقل عن البرنامج الأصلي ، وعندما يراد حساب تلك الدالة في البرنامج الأصلي يتم استدعاؤها عن طريق تعليمية تنفيذية تشمل في طرفها الأيمن اسم الدالة والمتغيرات اللازمة لحساب قيمتها .

وعندما ينتقل الحاسب إلى البرنامج الجزئي لحساب قيمة الدالة فإنه يعود إلى البرنامج الأصلي ، إلى التعليمة التي تم استدعاؤه منها عن طريق التعليمة RETURN Statement .

والصورة العامة للبرامج الجزئية التي تستخدم لحساب دالة هي :

FUNCTION name ( $a_1, a_2, \dots, a_n$ )
:
:
name = expression
RETURN
END

حيث : FUNCTION : كلمة تعريف للحاسب بأن التعليمات التالية تمثل برنامج جزئي لحساب دالة ما ،

name : اسم للدالة المطلوب حسابها ، وقيمة الدالة من حيث أنها قيمة حقيقية أو صحيحة يتوقف على اسم الدالة ، ( $a_1, a_2, \dots, a_n$ ) : متغيرات تلزم لحساب قيمة الدالة ، فإذا كان إحداها يمثل مصفوفة فإنه يجب وضع تلك المصفوفة في جملة توضيحية DIMENSION أو REAL أو INTEGER قبل أي جملة تنفيذية في البرنامج ،

RETURN : تعليمية للعودة إلى البرنامج الأصلي إلى الموضع الذي تم استدعاء الدالة منه ،

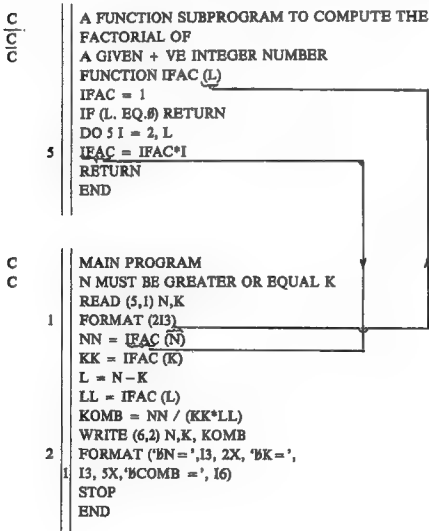
END : تعليمية نهاية البرنامج الجزئي .

## مثال (٩) :

كما ذكرنا في مقدمة هذا الفصل ، فانه لحساب التوافيق  $C_R^N$  بين عددين صحيحين  $R, N$  فان ذلك يتطلب حساب المضروب ثلاث مرات هي  $N!, R!, (N-R)!$  حيث أن :

$$C_R^N = \frac{N!}{R! (N-R)!} .$$

ولذا فمن المفيد في هذه الحالة أن نكتب برنامجاً جزئياً لحساب مضروب أي عدد صحيح . وفي البرنامج الأصلي نستدعي ذلك البرنامج الجزئي كلما تطلب الأمر حساب مضروب عدد ما .



نلاحظ في المثال السابق ما يلي :

١ - اعطينا البرنامج الجزئي الذي سيقوم بحساب دالة المضروب اسم IFAC وذلك كي نضمن أن النتيجة ستكون قيمة صحيحة لا تحتوي على أية كمور .

٢ - في البرنامج الأصلي ، يتم استدعاء البرنامج الجزئي عن طريق تعليمة تنفيذية يذكر في طرفها الأيمن اسم الدالة والمتغيرات الداخلة فيها ، وليس من المهم أن تكون أسماء تلك المتغيرات هي نفس أسماء متغيرات دالة البرنامج الجزئي ولكن المهم أن تكون من نفس النوع ( صحيحة أو حقيقية ) .

٣ - أن البرنامج الجزئي الذي سيقوم بحساب الدالة قد يحوي أكثر من تعليمة RETURN ، ولكنه يحوي تعليمة انتهاء البرنامج END مرة واحدة .

٤ - أن قيمة الدالة يتم حسابها في البرنامج الجزئي عن طريق تعليمة حسابية يكون الطرف الأيسر فيها هو اسم الدالة .

مثال (٧) :

لنفترض أن لدينا ٢٠ مجموعة من الأعداد وكل مجموعة تتكون من ٥ أعداد قد تحتوي على أعداد سالبة . ويراد حساب الوسط الحسابي لكل مجموعة تحوي ٥ أعداد جميعها موجبة . وسنفترض أن البرنامج الرئيسي سيقوم بقراءة مجموعات الأعداد وكتابة الوسط الحسابي للمجموعة التي تفي بالشروط. الموضوع . كما سنفترض أن هناك برنامجين جزئيين لدالتين احدهما تسمى TEST وذلك لأختبار أعداد كل مجموعة وهل جميعها أكبر من الصفر أم لا ، فإن كانت المجموعة تحوي عدداً سالباً على الأقل فإن قيمة تلك الدالة ستساوي صفراً ، بينما الدالة الأخرى وتسمى AVR وتختص بحساب الوسط الحسابي للمجموعة التي تفي بالشروط .

C	MAIN PROGRAM						
	DIMENSION GROUP (5)	<table><tr><td>3.</td><td>2.</td><td>-4.</td><td>5.</td><td>7.</td></tr></table> 1	3.	2.	-4.	5.	7.
3.	2.	-4.	5.	7.			
	REAL MEAN						
	DO 10 I=1, 20						
	READ (2,5) (GROUP (J) , J=1, 5)	<table><tr><td>6.</td><td>3.</td><td>5.</td><td>1.</td><td>4.</td></tr></table> 2	6.	3.	5.	1.	4.
6.	3.	5.	1.	4.			
5	FORMAT (5F6.2)						
	MEAN = TEST (GROUP)	:					
10	WRITE (6,15) (GROUP (J) , J=1, 5), MEAN:	:					
15	FORMAT (5F7.2, 5X, 'AVERAGE =', F7.2)	:					
	STOP						
	END	<table><tr><td></td><td></td><td></td><td></td><td></td></tr></table> 20					

```

C      A FUNCTION SUBPROGRAM TO TEST A GROUP OF NOS.
      FUNCTION TEST (ARRAY)
      DIMENSION ARRAY (5)
      TEST = 0.0
      DO 10 K = 1,5
      IF (ARRAY (K). LT. 0.0) RETURN
10     CONTINUE
      TEST = AVR (ARRAY)
      RETURN
      END

```

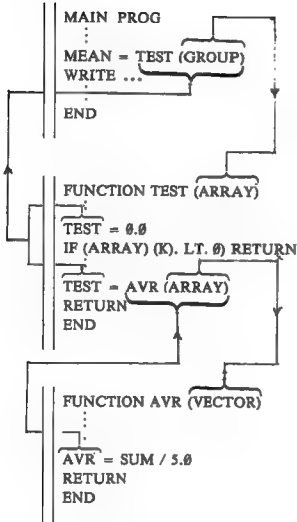
```

C      FUNCTION SUBPROGRAM TO COMPUTE AVERAGE
      FUNCTION AVR (VECTOR)
      DIMENSION VECTOR (5)
      SUM = 0.0
      DO 5 I = 1,5
5     SUM = SUM + VECTOR (I)
      AVR = SUM / 5.0
      RETURN
      END

```

في هذا المثال نلاحظ أن :

١ - البرنامج الأصلي يستدعي برنامجاً جزئياً لحساب الدالة TEST ، وأن البرنامج الذي يقوم بحساب تلك الدالة يقوم بإستدعاء برنامجاً جزئياً آخر لحساب الدالة AVR ، ولذا فان تسلسل العمليات سيكون كالتالي ( أنظر الشكل ) .



(أ) قراءة مجموعة الأعداد عن طريق البرنامج الأصلي والمخرونة في المصفوفة **GROUP** .

(ب) الانتقال بتلك الأعداد الى البرنامج الجزئي للدالة **TEST** ،

(ج) في البرنامج الجزئي للدالة **TEST** يتم التعامل مع مجموعة الأعداد في مصفوفة تحت اسم **ARRAY** ، ويُجرى اختبار على مجموعة الأعداد تلك فان كانت تحتوي أية أعداد سالبة فان الدالة تأخذ القيمة صفراً ثم تنتقل الحسابات الى البرنامج الأصلي لكي يتم طباعة مجموعة الأعداد تلك وقيمة الدالة **TEST** ، أما اذا كانت جميع الأعداد موجبة فان الحسابات تنتقل الى البرنامج الجزئي للدالة **AVR** حيث يعبر اسم الدالة عن قيمة الوسط الحسابي لتلك المجموعة من الأعداد التي يتم التعامل معها في مصفوفة أخرى تحت اسم **VECTOR** .

- (د) بعد أن يتم حساب قيمة الدالة AVR ، تنتقل الحسابات مرة أخرى إلى البرنامج الجزئي للدالة TEST كي تكون قيمة تلك الدالة معبرة عن قيمة الوسط الحسابي لمجموعة الأعداد المعطاة .
- (هـ) في نهاية البرنامج الجزئي للدالة TEST تعود الحسابات مرة أخرى إلى البرنامج الرئيسي والذي يتم فيه وضع قيمة الدالة TEST (الوسط الحسابي) في المتغير MEAN .
- (و) في نهاية البرنامج الرئيسي يتم كتابة مجموعة الأعداد وقيمة وسطها الحسابي ، ثم يتوقف البرنامج الرئيسي عن الحساب .

٢ - البرنامجين الجزئيين للدالتين تحويان في كل منهما التعليمة الوصفية DIMENSION ، وأن عدد... الحللا المخصصة لكل مصفوفة في الدالتين يساوي العدد المخصص للمصفوفة GPOUP وأن اسم المصفوفة بدون وصف أو بُعد هو الذي يظهر كأحد متغيرات الدالة .

٣ - ومع أن البرنامج الأصلي مرتبط مع البرنامجين الجزئيين للدالتين ، إلا أن كل برنامج من البرامج الثلاث يمكن اعتباره برنامجاً مستقلاً بأسماء المتغيرات وأرقام التعليمات التي فيه وكلاً منها يحتوي على تعليمة النهاية END .

٤ - ان النتيجة النهائية لكل برنامج جزئي لحساب دالة عبارة عن قيمة واحدة فقط وليس أكثر تمثل قيمة تلك الدالة ، وقد يكون هذا هو أحد الاختلافات الجوهرية بين البرامج الجزئية للدوال والبرامج الجزئية الفرعية SUBROUTINE Subprograms التي ستحدث عنها الآن .

#### ٨-٤ البرامج الجزئية الفرعية SUBROUTINE Subprograms

احدى الوسائل الأخرى الفعالة في اختصار برنامج يحتوي على مجموعة من التعليمات الحسابية المتكررة هي استخدام ما يعرف بالبرامج الجزئية الفرعية . وهناك بعض أوجه الشبه بين هذا النوع من البرامج وبين البرامج الجزئية للدوال ، منها :

١ - أن كلاً منهما يعتبر برنامجاً مستقلاً عن البرنامج الأصلي ، في أسماء متغيراته وأرقام التعليمات التي يحتوي عليها ولا يربط بينهما وبين البرنامج الأصلي سوى ورود اسم البرنامج الجزئي في صورة تعليمة في البرنامج الأصلي .

٢ - أن كلاً من النوعين يحمل اسماً يميزه عن بقية البرامج الجزئية الأخرى ، كما أنه عن طريقه يتم استدعاؤه إلى البرنامج الأصلي . والصورة العامة للتعليمة الأولى في البرامج الجزئية الفرعية هي :

SUBROUTINE name	أو
SUBROUTINE name (a <sub>1</sub> , a <sub>2</sub> , ..., a <sub>n</sub> )	

والشروط الموضوعية على إعطاء الاسم للبرنامج الجزئي الفرعي هي نفس الشروط بالنسبة لأي متغير ، أي أن الاسم لا يزيد عن ستة حروف وأرقام بحيث يكون أوله حرفاً . والاسم هنا لا يدل على نوع المتغير التي سيكلف البرنامج الجزئي الفرعي بحسابها ، فقد يكون الاسم صحيحاً Integer بينما تكون النتيجة في صورة أعداد حقيقية . وقد تكون هذه إحدى فقط الاختلاف بين البرامج الجزئية للدوال والبرامج الجزئية الفرعية .

٣ - أن كلاً منهما يحتوي بداخله على تعليمة عودة RETURN Statement أو أكثر إلى البرنامج الأصلي ، كما أن كلاً منهما يحتوي على تعليمة نهاية END Statement واحدة لانتهاء البرنامج الجزئي . وبالتالي فإن الصورة العامة للبرامج الجزئية الفرعية ككل هي :

SUBROUTINE name ( $a_1, a_2, \dots, a_n$ ) : RETURN } END	مجموعة تعليمات بلغة الفورتران
--	-------------------------------

حيث : name : اسم البرنامج الجزئي الفرعي ، ولا يشترط هنا أن يكون الاسم دالاً على نوع النتائج ( حقيقية أم صحيحة ) التي سيقوم بحسابها ، كما لا يجب استخدام اسم البرنامج الجزئي الفرعي كمتغير في البرنامج الأصلي .

( $a_1, a_2, \dots, a_n$ ) : متغيرات يتم استخدامها في البرنامج الجزئي الفرعي وبعض تلك المتغيرات يكون قد سبق إعطاؤها قيمة في البرنامج الأصلي وبعضها الآخر قد يستخدم كنتاج لحسابات البرنامج الجزئي الفرعي ، كما أن بعض تلك المتغيرات قد تمثل مصفوفات ذات بعد واحد أو بعدين . فمثلاً لنفرض أن البرنامج الجزئي الفرعي يقوم بحساب مجموع مصفوفتين  $Y, X$  كلاً منهما ذات بعدين ( $M \times N$ ) ، فإن التعليمة الأولى في البرنامج الفرعي ستكون :

SUBROUTINE MATADD ( $M, N, X, Y, Z$ ) DIMENSION X ( $M, N$ ), Y ( $M, N$ ), Z ( $M, N$ )
---

وذلك بافتراض أن المصفوفة  $Z$  هي التي سيتم فيها جمع المصفوفتين  $Y, X$  .

وكما في حالة البرامج الجزئية للدوال ، يجب تعريف أبعاد ثلاث مصفوفات مختلفة في البرنامج الأصلي ، وكل منها ذات بعدين يساوي  $N, M$  ، ولا يهم أن كانت اسمائها هي نفس أسماء مصفوفات البرنامج الجزئي الفرعي أم لا . كذلك قد لا يحتاج البرنامج الجزئي الفرعي إلى أية متغيرات . وعلى سبيل المثال ، فإن البرنامج الجزئي الفرعي التالي لا يقوم مثلاً إلا بكتابة عنوان ما كلما يتم استدعاؤه عن طريق البرنامج الأصلي ، ولذا لا يتطلب أية متغيرات لعمل أية حسابات .



```

SUBROUTINE TITLE
WRITE (6,5)
FORMAT ('1', 30X, 'ADDITION OF TWO MATRICES' / 30X, 24 (1H=) //)
5 RETURN
END

```

تتمثل أية تعليمات بلغة الفورتران سواء كانت تلك التعليمات حسابية أو منطقية أو تعليمات انتقال .. الخ . ولكن يشترط ألا تكون بين تلك التعليمات تعليمات FUNCTION, SUBROUTINE . وقد يمكن من خلال تعليمات البرنامج الجزئي الفرعي استدعاء برنامج جزئي لحساب دالة أو برنامج جزئي فرعي آخر .

**RETURN :** تعليمة العودة الى البرنامج (\*) الذي تم منه استدعاء البرنامج الجزئي الفرعي . وقد يجتري البرنامج الجزئي الفرعي على أكثر من تعليمة من هذا النوع .

**END :** تعليمة انتهاء البرنامج الجزئي الفرعي .

#### ٨-٤-١ : تعليمة استدعاء البرنامج الجزئي الفرعي : CALL STATEMENT

في الأنواع السابقة من الدوال ( الدوال سابقة التخزين أو الدوال في صورة تعبير رياضي أو البرامج الجزئية للدوال ) ، كان يتم استدعاؤها عن طريق وضع اسمها في شكل تعليمة حسابية مثل :

$$X = 5.0 * \text{SQRT} (A^{**}2 + B^{**}2)$$

$R1 = \text{ROOT} (A, B, C)$

حيث ROOT يمثل تعبير رياضي لدالة يتم وضعها في أول البرنامج الأصلي أو يمثل برنامج جزئي لدالة تقوم بحساب جذر معادلة من الدرجة الثانية معاملاتها  $C, B, A$  مثلاً . ويختلف الحال بعض الشيء في البرامج الجزئية الفرعية حيث يتم استدعاء هذا النوع من البرامج عن طريق التعليمة CALL . والصورة العامة لتلك التعليمة هي :

CALL name (a<sub>1</sub>, a<sub>2</sub>, ..., a<sub>n</sub>)

حيث name : اسم البرنامج الجزئي الفرعي المطلوب الانتقال إليه .

a<sub>1</sub>, a<sub>2</sub>, ..., a<sub>n</sub> : متغيرات سبق معرفة قيمها وقد تكون ثوابت أيضاً حقيقية أو صحيحة ، كما قد يمثل أحد تلك المتغيرات أو بعضها النتيجة أو النتائج التي قام البرنامج الجزئي الفرعي بحسابها

(٥) قد يكون هذا البرنامج هو البرنامج الأصلي أو برنامج جزئي لدالة أو برنامج جزئي فرعي آخر .

تمهيداً للعودة بتلك النتائج الى الموضع الذي تم منه استدعاء البرنامج الجزئي الفرعي . فعلى سبيل المثال يمكن أن تكون صيغة الاستدعاء للبرنامج الجزئي الفرعي ROOT الذي يقوم بحساب جذري معادلة من الدرجة الثانية معاملاتها هي C,B,A في الصورة :

CALL ROOT (A,B,C ROOT1, ROOT2)

وفي هذه الحالة يجب أن تكون قيم المتغيرات C,B,A معروفة مسبقاً ، وقبل أن يتم استدعاء البرنامج الجزئي الفرعي ROOT . ولكن ما هي المتغيرات ROOT1, ROOT2 التي ظهرت في تعليمة الاستدعاء ؟ والاجابة على ذلك هي نتائج الحسابات المطلوبة من البرنامج الجزئي الفرعي أن يقوم بها .

وعند هذه النقطة ، فهناك اختلافين جوهريين بين البرامج الجزئية الفرعية وبقية الأنواع الأخرى التي سبق دراستها .

١ - ففي البرامج الجزئية الفرعية يمكن حساب قيم أكثر من متغير ( ROOT1, ROOT2 مثلاً ) والعودة بها الى التعليمة التي تم منها استدعاء البرنامج الجزئي الفرعي ، بينما الأنواع الأخرى سواء البرامج الجزئية للدوال أو الدوال التي في صورة تعبير رياضي لا يمكنها حساب أكثر من قيمة واحدة .

٢ - في البرامج الجزئية الفرعية يتم الحصول على النتائج من اسماء المتغيرات التي حسبت فيها تلك النتائج ، بينما في الأنواع الأخرى فان اسم الدالة هو الذي يتم فيه الاحتفاظ بالنتيجة .

وعطفاً للخاصية الأولى ، فلا نستطيع مثلاً استخدام برنامج جزئي لدالة لحساب مصفوفة ، حيث أن المصفوفة تحوي أكثر من قيمة بينما يمكن عمل ذلك باستخدام البرامج الجزئية الفرعية . وفيما يتعلق بالخاصية الثانية ، فانه نظراً الى أن نتائج حسابات البرامج الجزئية الفرعية يتم تخزينها في متغيرات خاصة بتلك البرامج الجزئية ، الا أنه عند العودة الى البرنامج الأصلي أو الى النقطة التي تم منها الاستدعاء فان تلك النتائج تُفَرِّغ في المتغيرات المناظرة لها في تعليمة الاستدعاء CALL فعلى سبيل المثال ، اذا كانت تعليمة الاستدعاء هي :

CALL ROOT (A,B,C, ROOT1, ROOT2)

فهذا لا يؤثر بشيء اذا كانت متغيرات البرنامج الجزئي الفرعي كالتالي :

SUBROUTINE ROOT (X,Y,Z, R1, R2)

وفي هذه الحالة ، اذا أُريد طباعة قيمتي الجذرين في البرنامج الأصلي فيجب طباعة ROOT1 و ROOT2 وليس R1, R2 . والمهم في هذا الموضوع أن تكون عدد المتغيرات ونوعها ( صحيح أو حقيقي ) في تعليمة الاستدعاء يتطابق تماماً عدد المتغيرات ونوعها المناظرة لها في البرنامج الجزئي الفرعي ، ولا يهم ان كانت اسماء المتغيرات متشابهة أم لا .

مثال (١) :

```

SUBROUTINE ROOT (X,Y,Z,T1, T2)
R = SQRT (Y*Y-4.0 *X*Z)
IF (R) 3,2,1
1  S = 2.0*X
   T1 = (-Y+R) / S
   T2 = (-Y-R) / S
   RETURN
2  T1 = -Y / (2.0*X)
   T2 = T1
   RETURN
3  WRITE (6,4)
4  FORMAT ('IMAGINARY ROOTS')
   T1 = 9999.9
   T2 = T1
   RETURN
END

C  MAIN PROGRAM
   A = 2.0
   B = 4.0
   C = 1.0
   CALL ROOT (A,B,C, ROOT1, ROOT2)
   WRITE (6,1) ROOT1, ROOT2
1  FORMAT (2F14.4)
   STOP
   END

```

نلاحظ في هذا المثال ما يلي :

- ١ - في البرنامج الجزئي الفرعي ، هناك أكثر من تعليمة عودة RETURN الى البرنامج الأصلي .
- ٢ - ان هناك تعليمات طباعة يمكن ان يحتويها البرنامج الجزئي الفرعي .
- ٣ - ان اسماء المتغيرات التي احتوتها تعليمة الاستدعاء في البرنامج الأصلي تختلف عن اسماء المتغيرات في البرنامج الجزئي الفرعي ولكن بشرط أن عدد ونوع تلك المتغيرات واحدة في البرنامجين .

٤ - ان نتائج البرنامج الجزئي الفرعي ستحسب في المتغيرات T1, T2 بينما في البرنامج الأصلي مستخدم نظريتهما ROOT1, ROOT2 ، وعموماً فقد تتفق أو تختلف اسماء المتغيرات في البرنامجين فلا يهم هذا طالما أننا سبق أن اتفقتنا أن كلا من البرنامجين يعتبر مستقلاً عن الآخر .

٥ - في البرنامج الأصلي يمكن تغيير تعليمة الاستدعاء الى :

CALL ROOT (2.0, 4.0, 1.0, ROOT1, ROOT2)

أي أننا نضع قيم المتغيرات بدلاً من اسمائها ولكن لا يمكن عمل ذلك في البرنامج الجزئي الفرعي والا لكان برنامجاً جزئياً فرعياً خاصاً ، وهذا ما لا يتفق مع طبيعة استخدام مثل تلك البرامج الجزئية .

٦ - أن المتغيرات التي تمثل النتائج ، يتم كتابتها بعد بقية المتغيرات .

مسائل (٢) :

لنفرض أن هناك مصفوفتين X, Y يتم حسابهما داخل البرنامج الأصلي ، ويراد جمعهما اذا كانت جميع عناصرهما موجبة وغير صفرية . وسنستخدم برنامجاً جزئياً لدالة (TEST) لاختبار قيم المصفوفتين ، بينما سنستخدم برنامجاً جزئياً فرعياً (ADDMAT) لجمع هاتين المصفوفتين في حالة ما اذا تحقق الشرط السابق . وسنفترض أن عملية الجمع ستم في احدى المصفوفتين ولكن Y .

C	MAIN PROGRAM DIMENSION X (20, 10), Y(20,10) : DO 10 I = 1,M DO 10 J = 1,N : X (I,J) = ..... 10 Y (I,J) = ..... CALL ADDMAT (M,N,X,Y,IN) IF (IN.EQ.0) GO TO 30 DO 15 I = 1,M 15 WRITE (6,20) (Y(I,J), J = 1,N) 20 FORMAT (IX, 10 F 12.3) 30 STOP END
---	---

C	A SUBROUTINE SUBPROG. TO ADD TWO MATRICES. SUBROUTINE ADDMAT (M,N,X,Y,K) DIMENSION X(1,1), Y(1,1) K = ITEST (M,N,X,Y)
C	TESTING THE POSSIBILITY TO ADD OR NOT. IF (K.EQ.0) RETURN DO 10 I = 1, M DO 10 J = 1, N
10	Y(I,J) = Y(I,J) + X(I,J) RETURN END
C	A FUNCTION SUBPROG. TO TEST THE NUMBERS FUNCTION ITEST (M,N,X,Y) DIMENSION X(1,1), Y(1,1) ITEST = 0 DO 5 I = 1,M DO 5 J = 1,N IF (X(I,J). LT. 0.0) RETURN IF (Y(I,J). LT. 0.0) RETURN
5	CONTINUE ITEST = 1 RETURN END

نلاحظ في المثال السابق ما يلي :

- ١ - أن البرنامج الرئيسي يعتمد في تنفيذه على برنامج جزئي فرعي ADDMAT ، كما أن الأخير يحتاج في تنفيذه الى برنامج جزئي لدالة ITEST (أو برنامج جزئي فرعي آخر) .
- ٢ - أننا نتعامل مع مصفوفات في جميع البرامج وهذا يتطلب ذكر تعليمة DIMENSION لتحديد أبعاد تلك المصفوفات في تلك البرامج ، وفي البرنامجين الجزئيين اكتفينا بتحديد أبعاد المصفوفتين بالمتغيرات X(1,1), Y(1,1) . ولكن في البرنامج الرئيسي لابد من تحديد أبعادهما بالأرقام الفعلية كما أنه في البرنامج الجزئية لا يجب أن تتعدى تلك الأرقام (  $N \leq 10$  ), (  $M \leq 20$  ) .
- ٣ - بعد الانتهاء من تنفيذ البرنامج الجزئي الفرعي ADDMAT والعودة الى نقطة الاستدعاء في البرنامج الرئيسي ، فإن عملية جمع المصفوفتين من عدمها يتوقف على قيمة المتغير K في البرنامج الجزئي الفرعي وهي نفس القيمة للمتغير IN في البرنامج الرئيسي .
- ٤ - ان المتغيرات المستخدمة في حساب البرامج الجزئية عامة قد تكون جميعها من نوع واحد ( حقيقية أو صحيحة ) ، وقد تكون خليط منها .

### ٨-٤-٢ الفروق بين البرامج الجزئية للدالة والبرامج الجزئية الفرعية :

ولعدم التداخل في قواعد وكيفية استخدام البرامج الجزئية للدالة والبرامج الجزئية الفرعية ، سنلخص في الجدول التالي أهم الفروق بينهما .

برامج جزئية	لدالة FUNCTION	فرعية SUBROUTINE
اسم البرنامج الجزئي .	يتكون من ٦-١ حروف وأرقام بحيث يبدأ بحرف ولا يحتوي على علامات خاصة .	يتكون من ٦-١ حروف وأرقام بحيث يبدأ بحرف ولا يحتوي على علامات خاصة .
عدد المتغيرات المستخدمة	متغير أو أكثر .	بلون متغيرات أو بأكثر من متغير .
عدد النتائج التي يمكن العودة بها بعد التنفيذ .	نتيجة واحدة تحملها اسم الدالة .	قد لا يعود بأية نتائج ، وقد يعود بأكثر من نتيجة .
طريقة استدعاء البرنامج الجزئي .	بوضع اسم الدالة ومتغيراتها في صورة تعليمة رياضية .	باستخدام تعليمة الاستدعاء CALL .
العلاقة بين الاسم والقيمة ( أو القيم المحسوبة )	هناك علاقة ، فنوع النتيجة ( حقيقية أم صحيحة ) يتوقف على اسم الدالة .	ليست هناك علاقة .

وقد يتضح من الجدول السابق ، أن البرامج الجزئية الفرعية تعتبر أكثر مرونة في استعمالها ، ومع ذلك فقد يتطلب الأمر في بعض الأحيان ضرورة استخدام أحد النوعين دون النوع الآخر ، وهذا يتوقف على مهارة وخبرة واضع البرنامج .

## ٨-٥ تعليمة التعميم : COMMON Statement

### مقدمة :

فيما سبق عرفنا أن وسيلة الترابط والمشاركة بين البرنامج الرئيسي والبرنامج الجزئي أو مجموعة البرامج الجزئية التي يتطلب استخدامها مع البرنامج الرئيسي وكذلك بين البرامج الجزئية نفسها تتم عن طريق مجموعة من المتغيرات تظهر في كل من تعليمة الاستدعاء والتي توجد في البرنامج الرئيسي والتعليمة التي تبين اسم البرنامج الجزئي . وعلى سبيل المثال وكما سبق أن أوضحنا فإن تعليمة الاستدعاء في البرنامج الرئيسي يمكن أن تكون في الصورة :

CALL ADDMAT (M,N,X,Y,K)

وفي هذه الحالة فإن اسم البرنامج الجزئي قد يكون في الصورة :

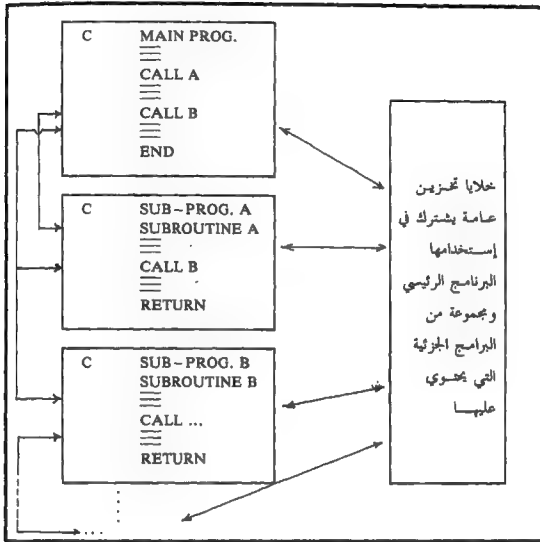
SUBROUTINE ADDMAT (NR,NC,A,B,KEY)

ولا يستلزم على المبرمج أن يستعمل أسماء متغيرات في البرنامج الجزئي تكون مختلفة عن تلك التي استخدمت في تعليمة الاستدعاء ، ولكن يتوجب أن يكون كل متغير في تعليمة الاستدعاء لها نظيرها وفي نفس الترتيب والنوع ( حقيقي أو صحيح ) في إسم البرنامج الجزئي .

وعندما يحتوي البرنامج الرئيسي على أكثر من برنامج جزئي وكل من تلك البرامج الجزئية يتم استدعاؤها عن طريق مجموعة كبيرة من المتغيرات تظهر في كل من تعليمة الاستدعاء - كما سبق أن بينا - وفي اسم البرنامج الجزئي فقد يكون من الأفضل استخدام تعليمة التعميم .

## تعليمة التعميم : COMMON Statement

تتم أهمية تلك التعليمة في تعميم ( تخصيص ) جزء من خلايا التخزين كي يتم استخدامها عن طريق البرنامج الرئيسي ومجموعة البرامج الجزئية التي يحتوي عليها : بمعنى آخر أن تلك التعليمة تستخدم في تعميم بعض خلايا وحدة التخزين في الحاسب ( أنظر الشكل ٨-١ ) كي يشترك في استخدامها كل من البرنامج الرئيسي وكذلك البرامج الجزئية ( كلها أو بعضها ) التي يتطلب استدعاؤها من البرنامج الرئيسي . أي أن تلك المنطقة مستخدمة للربط بين البرنامج الرئيسي وبرامجه الجزئية وكذلك بين البرامج الجزئية نفسها وذلك كبديل عن استخدام المتغيرات التي تظهر في تعليمة الاستدعاء والتعليمة التي تحتوي عليها اسم البرنامج الجزئي المطلوب استخدامه ، خاصة إذا كان عدد تلك المتغيرات كبيراً . ويجب أن تسبق تعليمة التعميم أي تعليمة تنفيذية أخرى في البرنامج الرئيسي أو البرامج الجزئية .



شكل (٨-١)

وهناك صورتين عامتين لتلك التعليلة :

COMMON List	أو
COMMON/Name/List	

حيث :

List : ترمز الى مجموعة المتغيرات ( صحيحة أو حقيقية - بسيطة أو ذات أبعاد ) التي سيتم تعميم وتخصيص خلايا تخزين لها لاستخدامها عند الضرورة في البرنامج الرئيسي وكذلك البرامج الجزئية .



أمثلة :

COMMON A,B,C,X (10)

COMMON R (6), J (3), S,T

ويتوقف عدد خلايا التخزين التي سيم تعميمها على عدد المتغيرات المستخدمة في التعليمات وأبعادها ( إن كانت ذات أبعاد ) . ويحتوي البرنامج الرئيسي على تعليمات تعميم واحدة من هذا النوع ، تشتمل على جميع المتغيرات التي ستستخدم في البرامج الجزئية المختلفة بحيث يوضع في كل برنامج جزئي تعليمات تعميم واحدة أيضا تشتمل نفس المتغيرات بترتيبها وأنواعها التي ظهرت بها في تعليمات التعميم الموجودة في البرنامج الرئيسي . ويمكن أن يوضع أيضا في كل برنامج جزئي تعليمات تعميم تحتوي على المتغيرات التي يتطلبها البرنامج الجزئي فقط دون غيرها من المتغيرات الباقية ، ولكن يجب الحذر هنا حيث أن عملية حجز قيم المتغيرات في تعليمات التعميم يتم بطريقة تناهية . فمثلاً إذا كانت تعليمات التعميم في البرنامج الرئيسي في الصورة :

COMMON A,D (5),B,C,X (2,3)

وكان هناك برنامجين جزئيين أحدهما يحتوي على تعليمات تعميم في الصورة :

COMMON A,B,X (2,3)

COMMON A,C,D (5)

والبرنامج الجزئي الآخر يحتوي على تعليمات التعميم :

فان مجموعة خلايا التخزين الخاصة بتعميم المتغيرات A (5), B,C,D (5), X (2,3) سيتم فيها حجز قيم المتغيرات السابقة بنفس الترتيب الذي كتبت به في تعليمات التعميم في البرنامج الرئيسي . أي أن خلايا التخزين ستكون في الصورة :

1	2	3	4	5	6	7	8	9	10	1	12	13	14
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
A	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	B	C	X <sub>11</sub>	X <sub>21</sub>	X <sub>12</sub>	X <sub>22</sub>	X <sub>13</sub>	X <sub>23</sub>

وعند تنفيذ البرنامج الجزئي الأول فان قيم المتغيرات التي تحتوي عليها تعليمات التعميم في هذا البرنامج الجزئي ستكون كالتالي :

- A المتغير A سيأخذ القيمة المخزنة في الخلية ١ وهي نفس قيمة المتغير  
 D (1) المتغير B سيأخذ القيمة المخزنة في الخلية ٢ وهي قيمة المتغير  
 D (2) المتغير X (1,1) سيأخذ القيمة المخزنة في الخلية ٣ وهي قيمة المتغير  
 D (3) المتغير X (2,1) سيأخذ القيمة المخزنة في الخلية ٤ وهي قيمة المتغير  
 D (4) المتغير X (1,2) سيأخذ القيمة المخزنة في الخلية ٥ وهي قيمة المتغير

C المتغير X (2,3) سيأخذ القيمة المخزنة في الخلية ٨ وهي قيمة المتغير

ونفس الوضع سينشأ عند تنفيذ البرنامج الجزئي الثاني حيث أن :

A المتغير A سيأخذ القيمة المخزنة في الخلية ١ وهي نفس قيمة المتغير  
B,D (1) المتغير C سيأخذ القيمة المخزنة في الخلية ٢ وهي القيمة التي خصصت للمتغيرين  
X(1,1), D(2) المتغير D سيأخذ القيمة المخزنة في الخلية ٣ وهي القيمة التي خصصت للمتغيرين

X(1,3),B المتغير D (5) سيأخذ القيمة المخزنة في الخلية ٧ وهي القيمة التي خصصت للمتغيرين

أي أن بعض خلايا التخزين في تلك المنطقة سيحتوي كل منها على قيمة ستخصص لأكثر من متغير مثل الخلايا ٢ ، ٣ ، ... ، ٨ . ولذا فيجب الحذر عند وضع تعليمات تعميم في البرامج الجزئية تختلف في محتواها وترتيبها - المتغيرات بها عن تعليمات التعميم التي تحتوي عليها البرنامج الرئيسي .  
ويتبين من هذا المثال أن :

١ - أي برنامج أساسي أو جزئي يجب ألا يحتوي على أكثر من تعليمات تعميم واحدة من هذا النوع .

٢ - ليس من الضروري أن تكون عدد وحدات خلايا التخزين في تعليمات التعميم من هذا النوع متساوية ، ولكن عادة ما يكون عدد وحدات خلايا التخزين في تعليمات التعميم في البرنامج الرئيسي أكبر من أو يساوي عدد وحدات خلايا التخزين لأي تعليمات تعميم موجودة في برنامج جزئي .

٣ - تعليمات التعميم قد تحتوي متغيرات بسيطة أو ذات أبعاد وبالتالي فإن تعليمات التعميم يمكن أن تحمل محل التعليمات التوضيحية لأبعاد المتغير DIMENSION Statement أي أنه إذا أحتوت تعليمات التعميم على متغيرات ذات أبعاد فلا يجب في هذه الحالة ظهور أسماء تلك المتغيرات في التعليمات التوضيحية الأخرى مثل REAL, INTEGER وإذا تطلب الأمر توضيح نوع بعض المتغيرات ذات الأبعاد من حيث أنها حقيقية أو صحيحة فيمكن عمل ذلك في التعليمات التوضيحية على أن تظهر أسماء تلك المتغيرات في تعليمات التعميم كمتغيرات بسيطة وليست ذات أبعاد . فمثلا يمكن كتابة :

```
INTEGER X (5,5), D (5)
COMMON X,D
```

أو بطريقة أخرى :

```
COMMON IX (5,5), ID (5)
```

مثال :

باستخدام تعليمة بالتعميم اكتب برنامج جزئي لجمع مصفوفتين .  
سنفترض أن المصفوفتين هما A,B وأن نتيجة الجمع ستكون في احداهما ولكن المصفوفة A .

C	MAIN PROG
C	NR IS NO. OF ROWS, NC IS NO OF COLUMNS.
	COMMON A (15,15), B (15,15), NR, NC
	READ (5,*) NR, NC
	DO 6 I=1, NR
6	READ (5,*) (A(I,J), J=1,NC)
	DO 7 I=1, NR
7	READ (5,*) (B(I,J), J=1, NC)
	CALL MATADD
	DO 8 I=1, NR
8	WRITE (6,9) (A(I,J), J=1, NC)
9	FORMAT (1X,10 F12.3)
	STOP
	END

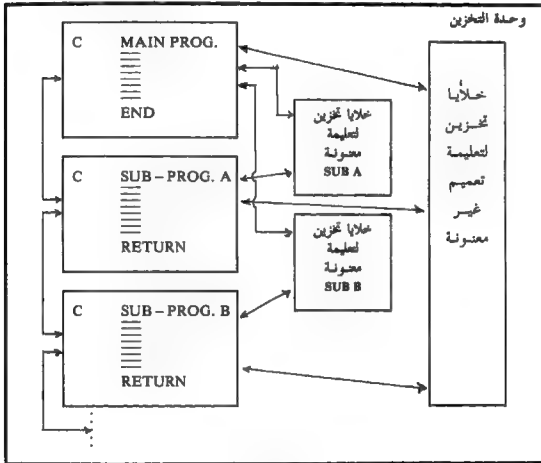
  

C	ASUBPROG TO ADD 2 MATRICES OF MAX. ORDER OF (15 × 15)
	COMMON X(15,15), Y(15,15), NR, NC
	DO 3 I=1, NR
	DO 3 J=1, NC
3	X(I,J) = X(I,J) + Y(I,J)
	RETURN
	END

أما الصورة الثانية لتعليمة التعميم فتختلف عن الأولى في أن كل تعليمة تعميم يصحبها اسم يميزها عن بقية تعليمات التعميم التي يحتوي عليها البرنامج الأصلي والبرامج الجزئية . وكل تعليمة تعميم من هذا النوع تحتوي على مجموعة من المتغيرات تستخدم في حساب برنامج جزئي ، وقد يشترك متغير أو أكثر في أكثر من تعليمة تعميم . كما أن الاسم الذي يميز تعليمة التعميم عن بقية تعليمات التعميم الأخرى ينطبق عليه نفس الشروط التي تنطبق على اسم أي متغير في لغة الفورتران ، أي يتكون من مجموعة من الحروف والأرقام لا يزيد عددها عن ستة وبشرط أن يبدأ بحرف وليس رقما ولا يحتوي على أية إشارات خاصة .

كما يلاحظ أن هذا النوع من تعليمات التعميم يبدأ بكلمة التعميم COMMON ثم يأتي اسم التعليمة بين علامتي قسمة (Slash) ثم تأتي مجموعة المتغيرات التي تخص بها تلك التعليمة . كما أن لكل تعليمة تعميم من هذا النوع مجموعة من خلايا التخزين تخص بتلك التعليمة دون غيرها ( أنظر الشكل ٨-٢ ) ، بحيث لا تتداخل خلايا تخزين تعليمة ما مع خلايا تخزين - تعليمة أخرى . ولذا فيمكن أن نطلق على هذا النوع من تعليمات التعميم اسم تعليمات التعميم المعنونة

#### Labeled COMMON Statement



شكل (٨-٢)

ومن الأفضل وضع المتغيرات التي لا تأخذ قيمة عديدة مثل المتغيرات المنطقية Logical Variables أي التي تأخذ إحدى القيمتين TRUE أو FALSE والمتغيرات الأخرى التي تمثل قيمها بحروف وليس بأرقام عديدة أن توضع مثل تلك المتغيرات في تعليمات تعميم معنونة منفصلة عن تعليمات التعميم الأخرى ، أي لا تحتوي تعليمة التعميم على خليط من - المتغيرات ذات القيم العديدة وغير العديدة .

والاستخدام الأمثل للمثل هذا النوع من تعليمات التعميم يكون عندما يتطلب الأمر وجود مجموعة من البرامج الجزئية بجانب البرنامج الأصلي ، وكل برنامج جزئي يتطلب لحسابه مجموعة من المتغيرات قد تختلف جزئياً أو كلياً عن مجموعة المتغيرات اللازمة لحساب البرنامج الجزئي الآخر . وفي هذا النوع من تعليمات التعميم يجب أن يكون عدد وحدات خلايا التخزين في تعليمة التعميم المعنونة والتي توجد في البرنامج الأصلي مساوية تماماً لعدد وحدات خلايا التخزين في تعليمة التعميم المعنونة والموجودة في البرنامج الجزئي . وقد يكون هذا أحد الفروق الأساسية بين تعليمات التعميم العادية وتعليمات التعميم المعنونة . كما يمكن جمع تعليمتي تعميم أو أكثر من هذا النوع في تعليمة تعميم واحدة .

فعل سبيل المثال ، بدلاً من كتابة :

COMMON / A / VAR1, VAR2, MAT (3,3)

COMMON / B / VAR3, VAR4, VAR5, ARRAY (8)

كلاً على حدة ، يمكن دمجهما في تعليمة تعميم واحدة في الصورة :

COMMON /A/ VAR1, VAR2, MAT (3,3) /B/ VAR3, VAR4, VAR5, ARRAY (8)

#### EQUIVALENCE Statement

#### ٨-٦ تعليمة التكافؤ :

##### مقدمة :

من المعروف أن لكل متغير خلية ( عنوانا في وحدة التخزين ) أو مجموعة من خلايا التخزين تنسب الى هذا المتغير وتحتفظ بقيمته فيه كي يستخدم كلما ورد اسم ذلك المتغير في البرنامج . وقد يكون من الغريب أن تخصص خلية تخزين لأكثر من متغير في نفس البرنامج ، ولنا أن نتوقع ان امكن تنفيذ ذلك - مدى ما ستوفره من خلايا تخزين قد تكون ذات فائدة ، خاصة اذا كان البرنامج كبيراً ويحتوي على مجموعة كبيرة من المتغيرات والتي قد يستعمل بعضها في اجزاء معينة من البرنامج دون الحاجة اليها في مواضع أخرى من نفس البرنامج ، مما يجعل مثل تلك المتغيرات تحتل مساحات في وحدة التخزين قد تكون في ميسر الحاجة اليها خاصة اذا كانت سعة وحدة التخزين في الحاسب محدودة . ولكن تعليمة التكافؤ « في لغة الفورتران » تسمح بتنفيذ ذلك بشرط الا يتواجد متغيرين أو مجموعة من المتغيرات تتقاسم خلية تخزين واحدة على يمين علامة التساوي في تعليمة تنفيذية واحدة في البرنامج . كما أن هناك نوع من الارتباط بين تعليمة التكافؤ وتعليمة التعميم التي سبق شرحها ، سندحاول توضيحه فيما بعد .

## ٨-٦-١ الصورة العامة لتعليلة التكافؤ :

تأخذ تعليلة التكافؤ الصورة العامة التالية :

EQUIVALENCE ( $V_1, V_2, \dots, V_m$ )
--

حيث  $V_1, V_2, \dots, V_m$  مجموعة متغيرات تحتل مكاناً واحداً في وحدة التخزين . وقد تكون بعض تلك المتغيرات بسيطاً وبعضها الآخر تمثل عناصر في مصفوفة . ويلاحظ أن مجموعة المتغيرات تلك توضع بين قوسين .

أمثلة :

EQUIVALENCE (A(3),X)

EQUIVALENCE (R,S,K,V)

ويمكن دمج أكثر من تعليلة تكافؤ في تعليلة واحدة . فعلى سبيل المثال يمكن كتابة تعليمتي التكافؤ السابقتين في تعليلة واحدة كالتالي :

EQUIVALENCE (A(3),X), (R,S,K,V)

وهناك بعض الشروط في استخدام تعليلة التكافؤ في أي برنامج ، منها :

- ١ - يجب أن تسبق تعليلة التكافؤ أي تعليلة تنفيذية في البرنامج وكذلك الدوال ذات التعبير الرياضي ، كما أنها يجب ألا تسبق التعليمات التوضيحية مثل REAL, INTEGER, DIMENSION
- ٢ - كل مجموعة متغيرات مراد أن يكون لها نفس خلية التخزين ، يجب أن تحتوي على متغيرين على الأقل .

- ٣ - لا يشترط أن تكون مجموعة المتغيرات المتكافئة من نوع واحد ، أي جميعها حقيقية أو صحيحة ولكن يمكن أن يكون بعضها حقيقي والآخر صحيح طالما أنه لا يوجد أكثر من متغير واحد من تلك المتغيرات على يمين علامة التساوي من تعليلة تنفيذية . فمثلاً :

EQUIVALENCE (DEGFHT, K)

READ (3,\*) DEGFHT

DEGCNT = DEGFHT - 32.

$$\vdots$$

$$K = I + J$$

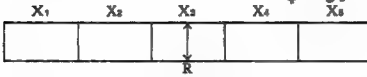
فالتغير DEGFHT ( فهرستية ) لن يستخدم مرة أخرى في البرنامج بعد تحويله الى DEGCNT ( درجات مئوية ) ، وطالما أن تلك التعليمة لا تحتوي على المتغيرين K, DEGFHT معا .

٤ - في تعليمة التكافؤ بين متغير ومصفوفة يوضع عنصر من المصفوفة كأحد المتغيرات ولا توضع المصفوفة كلها ، الا اذا كان التكافؤ بين مصفوفتين في جميع عناصرهما . فمثلاً اذا كانت R تمثل متغير ما وكانت X تمثل مصفوفة تحتوي على خمس قيم ، فلا يمكن كتابة تعليمة التكافؤ التالية :  
EQUIVALENCE (X, R)

ولكن يمكن أن يمثل المتغير R وأحد عناصر المصفوفة X خلية ما في وحدة التخزين .

EQUIVALENCE (X(3), R) أي يمكن كتابة :

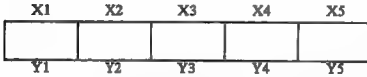
والتي تعني أن المتغير R يشترك مع العنصر الثالث من المصفوفة X في نفس خلية التخزين . ويمكن تمثيل ذلك في وحدة التخزين كالآتي :



بينما يمكن كتابة تعليمة التكافؤ بين عناصر المصفوفة X جميعها وعناصر مصفوفة أخرى Y ذات خمس قيم أيضاً بإستخدام التعليمة :

EQUIVALENCE (X,Y)

وفي هذه الحالة ستكون خلايا التخزين المشتركة في الصورة :



ونفس خلايا التخزين المشتركة السابقة يمكن تحقيقها بإستخدام تعليمة مثل :

EQUIVALENCE (X(1), Y(1))

EQUIVALENCE (X(2), Y(2))

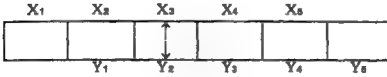
أو

وهكذا ، بحيث يكون الدليل العددي للمتغير X يتطابق مع الدليل العددي للمتغير Y وفي جميع الحالات السابقة يعتبر التكافؤ بين المصفوفتين كاملاً ، أي أن كل خلية من الخلايا الخمس تتجوز للمتغيرين .

ولكن اذا كانت تعليمة التكافؤ هي :

EQUIVALENCE (X(3), Y(2))

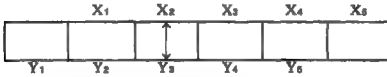
فإن مجموعة خلايا التخزين ستكون في الصورة :



واذا كانت تعليمة التكافؤ في الصورة :

EQUIVALENCE (X(2), Y(3))

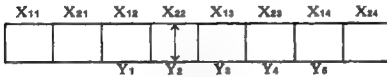
فستكون خلايا التخزين في الصورة :



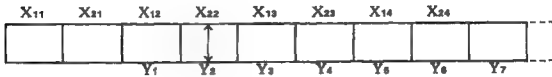
واذا افترضنا أن احدى المصفوفتين ولتكن  $X$  ذات بعدين  $X(2,4)$  ، فان تعليمة تكافؤ مثل :

EQUIVALENCE (Y(2), X(2,2))

تكون خلايا التخزين في تلك الحالة في الصورة :



واذا كانت المصفوفة  $Y$  تحتوي على ٥٠ قيمة مثلا ، فإن نفس التعليمة السابقة يمكن تمثيلها كالتالي :





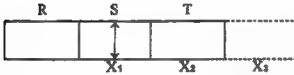
## ٨-٦-٢ الارتباط بين تعليمي التعميم والتكافؤ :

من الممكن ظهور اسم متغير بسيط أو ذو أبعاد في كل من تعليمي التعميم والتكافؤ في نفس البرنامج ، ولكن هناك بعض القيود التي يجب أخذها في الاعتبار حتى نتجنب المشاكل التي قد تنجم نتيجة لتواجد المتغير في التعليمتين مما قد ينتج عنه زيادة في منطقة التعميم قد يصرح بها أو لا يصرح بها .

فإذا أفترضنا الجزء التالي من البرنامج والذي يحتوي على تعليمة تعميم غير مُعنونة ،

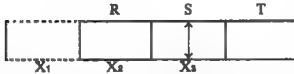
```
REAL X (3)
COMMON R,S,T
EQUIVALENCE (X(1), S)
```

فإن خلايا التخزين لهاتين التعليمتين ستكون كالتالي :



أي أننا سنضطر إلى إضافة بعض خلايا التخزين ( نحو اليمين ) في منطقة التعميم ، وهذا لا ينتج عنه مشاكل . ولكن إذا كانت تعليمة التكافؤ في جزء البرنامج السابق في الصورة :  
EQUIVALENCE (X(3), S)

فإن خلايا التخزين ستكون كالتالي :



وهذا يعني أننا نريد من الحاسب حجز خلية إضافية تسبق منطقة التعميم ، كي يوضع فيها العنصر X(1) وهذا غير مسموح به نظراً لأن عملية حجز قيم المتغيرات تتم بطريقة متتابعة مما يعني أن هذه الخلية الإضافية يكون الحاسب قد سبق حجزها لمتغير آخر خارج منطقة التعميم . وببساطة :

يمكن زيادة مساحة منطقة التعميم بالاتجاه نحو اليمين ولا يمكن ذلك بالاتجاه نحو اليسار .

- كذلك غير مسموح بكتابة تعليمة تكافؤ لمتغيرين (أو أكثر) بسيطين أو ذوي أبعاد إذا كان هذين المتغيرين سبق ذكرهما في تعليمة تعميم . فمثلاً من غير المنطقي كتابة الجزء التالي من برنامج .

COMMON U, V, W, X (3)

EQUIVALENCE (V, X(2))

حيث أنه في هذه الحالة فإن تعليمة التعميم ستخصص خلية تخزين لكل متغير أو عنصر في مصفوفة ، وبالتالي فإن تعليمة التكافؤ لن يمكن تحقيقها حيث أن خلايا التخزين ستكون في الصورة :

U	V	W	X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>

والتي يتضح منها أن أي متغيرين من متغيرات تعليمة التعميم لن يمكن جمعها في خلية تخزين واحدة .

- بالنسبة لتعليمات التعميم الممنونة والتي توجد في البرنامج الرئيسي واحد أو بعض البرامج الجزئية ، يجب أن يكون عدد خلايا التخزين متساوياً في تعليمات التعميم التي تحمل نفس الاسم . وبالتالي فإن إضافة أي خلايا تخزين أخرى لتلك التعليمة يجب أن يتم أيضاً في باقي تعليمات التعميم في البرامج الجزئية الأخرى .

## « تمارين »

١ - بين أي من التعليمات التالية يعتبر مسموحاً به ( كل تعليمة تعتبر مستقلة عن غيرها ) :

```
COMMON R, S, T(3,3), I,J
COMMON (X(2,2), ARRAY)
COMMON A, 6.2, B
COMMON X, Y, Z/LITS/A, B, C
COMMON/XYZ/X, Y, Z
EQUIVALENCE (X(5,5), Y(3,3))
EQUIVALENCE (X,Y,Z(2)), (B,L)
EQUIVALENCE (X(1), VAR, X(2), L)
EQUIVALENCE (A, 6.0, B)
```

٢ - أذكر قيمة المتغيرات الموجودة في كل من البرامج الجزئية التالية :

```
SUBROUTINE UN
DIMENSION A(3), C(2)
COMMON /XXX/ A,B /YYY/ C
A(2) = 1.0
C(1) = 1.0
RETURN
END
SUBROUTINE DEUX
DIMENSION B(3), C,D(2)
COMMON/XXX/C,B/YYY/D
B(2) = 2.0
C = 2.0
RETURN
END
```

```
SUBROUTINE TROIS
REAL X(4)
COMMON/XXX/X/YYY/A,B
X(1) = 2.0
X(2) = 2.0
X(3) = 2.0
X(4) = 2.0
A = 2.0
B = 2.0
RETURN
END
```

٣ - في البرنامج التالي ، يتم استدعاء البرامج الجزئية السابقة . أكتب قيم المتغيرات في تعليمة الطباعة .

```

1 | REAL X, Y, Z, R, S, T
   | COMMON / XXX / X,Y,Z,R / YYY / S,T
   | CALL TROIS
   | CALL DEUX
   | CALL UN
   | WRITE (7,1) X,Y,Z,R,S,T,
   | FORMAT (1X, 6E12.3)
   | STOP
   | END

```

٤ - بين أي من مجموعة التعليمات « أ » أو « ب » يعتبر مسموحاً به .

المجموعة « ب »

```

DIMENSION A (3), B(3)
COMMON / ZZ / A
EQUIVALENCE (A(1),B((2))

```

المجموعة « أ »

```

DIMENSION A(3),B(4),C(2)
COMMON / ZZ / A
EQUIVALENCE(A(2),B(2),C(1))

```

## « تمارين عامة »

١ - أعد صياغة التعبيرات التالية بلغة الفورتران :

$$\text{Radius} = \frac{a + b^h - 10^6}{\sqrt{(b^2 + 3a)}}; \text{Area} = \frac{314 R^3}{100 (a + b)}$$

$$b = \frac{-(x.y)^2 + (x.z)^2 - (y.z)}{x.y.z}; \text{Area} = \sqrt{S(S-a)(S-b)(S-c)}$$

$$\text{Diameter} = \frac{x^2 y^2 - Z^2}{xy - Z} - 1; \text{High} = \frac{1}{2} (x-y)^2 - \frac{1}{3} (y-z)^3 + \frac{1}{2} (z-x)^4$$

$$S = a + b + \frac{b-c}{d} \left( \frac{e^5}{f} \right); D = \frac{pb^3}{L^2} (3a + b)$$

$$\text{ArC} = 2 \sqrt{y^2 + \frac{4x^3}{3}}; R = \frac{x}{y-z} (3x^2 - 6x)$$

٢ - إذا كانت :

$$X=2, Y=3, Z=4, J=2, K=2, KK=7, L=-3$$

وضح أسبقية تنفيذ مكونات التعليمات التالية ، ثم احسب قيمتها :

$$a) S = -(X*Y) **3 + (X*Z) **2 - Y*Z / (X*Y*Z)$$

$$b) I = (J*(K-KK) / (9+L)); d) III = J*(K-KK) / (9+L)$$

$$c) II = (J*(K-KK)) / (9+L); e) IV = J* ((K-KK) / (9+L))$$

٣ - إذا كانت :

$$A=2, B=1., C=1., D=3., E=2., F=1., G=-1.$$

أوجد القيمة التي سيخزنها الحاسب في وحدة التخزين للمتغيرات التالية مع توضيح كيفية

خطوات الحساب :

$$Q = ((A/B*D) **2/D) **E$$

$$R = A*B/C*D/E*F$$

$$J = A + (B+C -(D*E+F))$$

$$T = A + B/C + D**E*F-G$$

٤ - إذا كانت :  $A=1., B=2., C=1., D=2., E=3., F=-1., G=1.$

أوجد القيمة التي سيخزنها الحاسب في وحدة التخزين للمتغيرات التالية ، مع توضيح كيفية خطوات الحساب :

$$Q = A*((B/C*D)/E) * F - G$$

$$J = ((A+B)/C) + D**E*(F-G)$$

$$S = A*((B+C) - (D*E) + F)$$

$$T = A/((D*D - G) **2/D) **E$$

$$LU = 0.5 * (A - B) **2 - (B - C) **3/3. + 0.5 * ((C - B)/D) **2$$

٥ - أعد كتابة التعليمات التالية في صورتها الصحيحة ( كل تعليمة مستقلة عن الأخرى ) .

```
DIMENSION X(100), Y(N), Z(M,10)
Z = Z + A*J
W(I) = W**2 + W(I,J)
Y = 2Y/ - A**0.5
A(I) = I + B - J/R
IF (X - 100) - 10,0,+100
GO TO K
READ ((7, 10) A(I,J), I=1,K, J=1,L)
GO TO I,(10,20,30,10)
DO 10 I=1,M,N,
```

٦ - في البرنامج التالي بعض الأخطاء ، أعد كتابته بصورة صحيحة :

```
51 A + 7D = C
R = 2.*L + C + B
READ = (7,I) A,B,L,D.
1 FORNAT (F8,3 - 2F6.2,13)
5 GO TO 5
IF (L - C) 7,5
7 WARITE (7.51) A,B,R - C
STOB
END
```

٧ - في البرنامج التالي توجد عدة أخطاء في لغة الفورتران ، أعد كتابة البرنامج بعد تصحيحها .

```

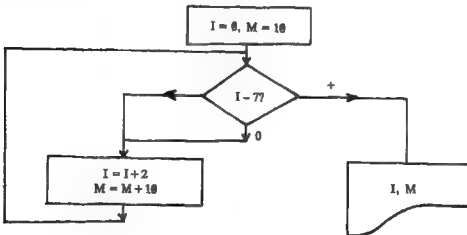
      READ (7,3,) A1,IA,A.B.
3    FORHAT (2F7,2 , F8.3/)
      IF (AB + AI - IA) 2,1
      DO 5 I=1, AI
2    IS = I + AI
      I = I + IS
5    CONTINUE
      GO TO (2,1,3), AI
1    IF (IS. NE. AB) GO TO 2
      A*I=AI
      WARITE (7,7)A*I,IA,AI
      STOB
      END
  
```

٨ - في البرنامج التالي بعض الأخطاء . أعد كتابته بصورة صحيحة .

```

8    READ (7,8) A,7B, KA, AK
      FORHAT (F6, 3, 15, 2I3)
      A + AK = KA
      WRIT (7,7) (R,B7,KA)
3    FOURMAT 15, F9. 12,6I
      STOP
      END
  
```

٩ - بافتراض مخطط التدفق التالي :



ماهي القيم النهائية للمتغيرات I ، M ؟

١٠- أعطيت مجموعة من البيانات عن عينة من طلبة وطالبات الجامعة عددها N بحيث أن كل بيان فيها يحتوي على :

١ - مستوى الطالب أو الطالبة حيث يوجد ٤ مستويات هي :

(LEVEL 1, LEVEL 2, LEVEL 3, LEVEL 4)

٢ - نوع الجنس ISEX التابع له هذا البيان ( = 1 اذا كان طالبا ، = 2 اذا كانت طالبة )

أرسم مخطط تدفق التعليمات لحساب :

١ - مجموع الطلبة والطالبات في كل مستوى من المستويات الأربعة في هذه العينة .

٢ - مجموع الطلبة في هذه العينة .

٣ - مجموع الطالبات في هذه العينة .

١١- باعت شركة الطيران السعودية هذا الشهر عدد N من التذاكر لركابها المتجهين لبلدة ما ، بعضها للأطفال الذين يقل أو يساوي عمرهم ١٠ سنوات والباقي لمن يزيد عن ذلك . فإذا كان سعر تذكرة الطفل هي K ريال ، سعر التذكرة للكبار هي L ريال ، أرسم مخطط تدفق تعليمات لحساب :

١ - حصيلة الشركة من تذاكر لأطفال .

٢ - حصيلة الشركة من تذاكر لكبار .

١٢- مصنع يحمل به N عامل وعاملة ، وكل منهم له حفيظة نفوس مبين فيها :

- النوع ( ١ اذا كان ذكرا ، ٢ اذا كانت انثى ) .

- السن ( من ١٨ سنة الى ٥٥ سنة ) .

- محل الميلاد ( ١ اذا كان في شمال المملكة ، ٢ اذا كان في شرق المملكة ، ٣ اذا كان في غرب المملكة ، ٤ اذا كان من مواليد جنوب المملكة ) .

أرسم مخطط تدفق التعليمات لحساب :

١ - عدد العاملين - عدد العاملات .

٢ - عدد العاملين والعاملات الذين يقل عمرهم عن ٣٠ سنة .

٣ - عدد العاملين والعاملات الذين ولدوا في كل منطقة من مناطق المملكة الأربعة .

١٣- تقوم شركة الكهرباء باستخراج فواتير شهرية مدون عليها رقم الاشتراك NFAC وقراءة العداد السابقة LREAD والقراءة الحالية NREAD وكذلك قيمة الاستهلاك . ويتم حساب



قيمة الاستهلاك بطرح قراءة العداد السابقة من القراءة الحالية . فإذا كان الفرق ( عدد الكيلوات - المستهلكة ) ID :

- حتى ١٠٠ كيلو يحسب الاستهلاك كالتالي :  $CONS1 = ID0 \times 0.08 + 10$
- مايزيد عن ١٠٠ وحتى ٤٠٠ كيلو وليكن عددها ID1 :  $CONS1 = ID1 \times 0.15 + 25$
- ما يتبقى فوق ٤٠٠ كيلو وات وليكن ID2 :  $CONS2 = ID2 \times 0.2 + 40$
- ( أي أن :  $ID = ID0 + ID1 + ID2$  ، قيمة الاستهلاك =  $CONS2 + CONS1 + CONS$  )

ارسم مخطط تدفق تعليمات لتلك البيانات مبتدئاً بقراءة البيانات اللازمة عن كل مشترك ( أو مستهلك ) ومتجهياً بكتابة الحسابات والبيانات الضرورية لكل فاتورة .

١٤- عدد العمال في مؤسسة ما هو N عامل ، بعضهم يعمل ٣٥ ساعة أسبوعياً أو أقل ويتقاضى كل من هؤلاء ٢٠ ريالاً عن كل ساعة . وآخرين يعملون أكثر من ذلك ويتقاضون ٢٥ ريالاً عن كل ساعة إضافية . ارسم مخطط تدفق تعليمات لحساب :

١ - عدد أفراد كل فئة . ٢ - مجموع ما تدفعه الشركة لكل فئة .

١٥- أعطيت مجموعة من الأعداد عددها N بعضها زوجي ( تقبل القسمة على ٢ بدون باق ) ، وبعضها الآخر فردي ( نتيجة القسمة تحتوي على باق ) . كما أن بعضها موجب والآخر سالبا .

ارسم مخطط تدفق تعليمات ( مع توضيح كيفية معرفة العدد الزوجي من الفردي ) لحساب :

- ١ - مجموع الأعداد الفردية SODD
- ٢ - مجموع الأعداد الزوجية SEVEN
- ٣ - مجموع الأعداد السالبة SNEG
- ٤ - مجموع الأعداد الموجبة SPOS

١٦- في أحد المصانع يتم حساب صافي دخل NETPAY كل عامل طبقاً للمعادلة التالية :

$$NETPAY = BP + OVT - ENCT - OTDED$$

حيث :

Base Pay	BP	تفصل الدخل الأساسي
Over Time	OVT	الأجر الإضافي
Income Tax	ENCT	ضريبة الدخل
Other Deduction	OTDED	خصومات أخرى

فإذا كان الدخل الأساسي BP الشهري لكل عامل يعتمد على عدد السنوات NY التي قضاها في المصنع بحيث أن :

$$\begin{array}{lll} \text{BP} = 800 * \text{NY} & \text{عندما} & (\text{NY} < 5) \\ \text{BP} = 1000 * \text{NY} & \gg & (5 \leq \text{NY} < 8) \\ \text{BP} = 1500 * \text{NY} & \gg & (\text{NY} > 8) \end{array}$$

ارسم مخطط تدفق تعليمات للخطوات التي سيتم بها حساب صافي دخل العاملين بذلك المصنع والذين يبلغ عددهم N ، مبتدئاً بقراءة البيانات اللازمة لكل عامل ثم كتابة بيانات العامل وصافي دخله .

١٧- عند إعطاء البرنامج التالي الى الحاسب ، ظهرت بعض الأخطاء في لغة الفورتران . أعد كتابتها في الصورة الصحيحة .

```

1  DIMENSION X (100.)
   READ (7,1) N
   FORMATE (20 13)
   READ (7,40) X(L), I=1, L)
   S = 0
   K = 0.
3  K = K + 1.
   IF (K. GT N.) GO TO 81
   S = S + (KX)
   GO TO 3.
18 R = S/N.
   WRITE (7,50) N, R.
50 FOURMAT (1X, THE NO. OF OBSERVATIONS IS. F14.8,5X,
   13 HVALUE OF R = 14.
   STOP
40 FORMAT (3(F6,2.2X)
   END

```

بعد اعادة كتابة البرنامج في صورته الصحيحة ، فان قيم X التي يتطلب قراءتها كانت كالتالي :

	1	2	3	4	5
X	13.22	31.40	12.08	21.18	22.12

أدخل تلك البيانات الى الحاسب بصيغة الادخال المعطاه في البرنامج أعلاه . وأكتب النتائج التي سيكتبها الحاسب طبقاً لصيغة الطباعة المعطاه في البرنامج .

١٨- عند اعطاء البرنامج التالي الى الحاسب ، ظهرت بعض الأخطاء في لغة الفورتران . أعد كتابتها في الصورة الصحيحة .

```

6      DIMENSION X1 (10), X2 (10), X3 (10.)
      READ (7,L) M
      FORMAT (3015)
      READ (7,10) XL (J), K=1,M),X2(1,I=1,M)
      L = 0
8      L = L + 1.
      IF ((L). GT. (M)) GO TO 10
      X3 (L) = X1. (L) + X5 (L)
      GO TO 8.
10     FORMAT (3(F4.1,2X)
100    WARITE (7.2) X3 (I), I=1-M
2      FORMAT (IX, THE 16. TH RESULT IS,/ 5X,3F6.1))
      STOP
      END

```

بعد اعادة كتابة البرنامج في صورته الصحيحة ، كانت قيم X2,X1 المطلوب اجراء العمليات الحسابية عليهما كالتالي :

	1	2	3	4	5
X1	-5.1	1.9	0.6	12.3	8.2
X2	6.1	8.2	-5.6	7.7	-3.2

أدخل تلك البيانات الى الحاسب بالصيغة المعطاه في البرنامج ، وأكتب النتائج التي ستحصل عليها من الحاسب طبقاً لصيغة الطباعة المعطاه في البرنامج .

١٩- في التمرين (١٦) أكتب البرنامج المناسب مستخدماً الجمل الشارحة المبينة للبيانات الداخلة حيث :

XX	يأخذ الصورة	NY	المتغير
XX.XX	يأخذ الصورة	OVT	المتغير
XX.XXX	يأخذ الصورة	ENCT	المتغير
XXX.XXX	يأخذ الصورة	OTDED	المتغير

٢٠- في المحررين (١٣) أكتب البرنامج المناسب مستخدماً تعليمات الادخال والطباعة المناسبة والجمل المناسبة اذا كان :

xxxxxx	يأخذ الصورة	LREAD	المتغير
xxxxxx	يأخذ الصورة	NREAD	والمتغير
xxxxxx a1 a2	يأخذ الصورة	NFAC	والمتغير
(حيث a2, a1 تمثل حروفا وليس أرقاماً)	يأخذ الصورة	CONS	والمتغير

٢١- اكتب برنامج لحساب :

$$SUM = \sum_{i=50}^{100} \frac{x_i^2}{i} - \sum_{i=1}^{20} x_i$$

٢٢- اكتب برنامج لقراءة مصفوفة غير خطية X عن طريق صفوفها ( صف في كل سطر ) ثم اعادة كتابتها بحيث يكتب العمود الأول في السطر الأول ، العمود الثاني في السطر الثاني وهكذا .

٢٣- لديك مجموعة من الأعداد :  $X_1, X_2, X_3, \dots, X_N$

أكتب برنامج لحساب الانحراف المعياري لها (STDEV) حيث :

$$VAR\ iance = \sum_{i=1}^N (x_i - \bar{x})^2 / N ; \bar{x} = \sum_{i=1}^N x_i / N ;$$

$$Standard\ DEV = \sqrt{VAR\ iance}$$

٢٤- اكتب برنامج لحساب الانحراف المعياري STDV لمجموعة من القيم  $X_i$  عددها N . مع العلم بأن الانحراف المعياري يمكن حسابه من المعادلة :

$$R^2 = \frac{\sum X_i^2}{N} - \left( \frac{\sum X_i}{N} \right)^2 ; STDV = \sqrt{R^2}$$

٢٥- مجموعتان من الأعداد عدد كل منها n ، سبق تخزينهما في مصفوفتين خطيتين X , Y . أكتب برنامج .

(أ) لحساب

$$\sum X_i^K Y_i \sum Y_i \sum X_i$$

وذلك باستخدام حلقات التنفيذ المتكررة

(ب) حساب ماسبق في الخطوة أ بدون استخدام حلقات التنفيذ المتكررة .

( K يمثل عدد صحيح موجب )

٢٦- بافتراض أن لديك مجموعة من الأعداد الموجبة عددها N ، تم حسابها وتخزينها في الحاسب في مصفوفة خطية X . أكتب برنامج :

( أ ) لاستخراج أكبر تلك الأعداد وموقعه بين مجموعة تلك الأعداد .

( ب ) لاستخراج أصغر تلك الأعداد وموقعه بين مجموعة تلك الأعداد .

( جـ ) لاعادة ترتيب تلك الأعداد ترتيباً تنازلياً أي أن :

$$x(1) \geq x(2) \geq x(3) \dots \geq x(N)$$

٢٧- في التخزين السابق ، وقبل اعادة ترتيب الأعداد تنازلياً ، أكتب برنامج لاستخراج أكبر خمسة أعداد في المصفوفة الخطية X ومواقعها .

٢٨- بافتراض أن  $A_{m \times n}$  تمثل مصفوفة عدد صفوفها M وعدد أعمدها N وتحتوي على مجموعة من الأعداد الموجبة ، أكتب برنامج :

١ - لقراءة تلك المصفوفة عن طريق أعمدها .

٢ - لاستخراج أكبر عنصر في تلك المصفوفة وموقعه ( في أي صف وأي عمود ) .

٣ - لاستخراج أصغر عنصر في تلك المصفوفة وموقعه ( في أي صف وأي عمود ) .

٤ - لاعادة ترتيب أعداد تلك المصفوفة ترتيباً تصاعدياً ، أي أن :

$$a_{1,1} \leq a_{1,2} \dots \leq a_{1,N} \leq a_{2,1} \leq a_{2,2} \dots \leq a_{2,N} \leq a_{3,1} \dots \leq a_{M,N}$$

٢٩- المصفوفة  $A_{M,M}$  تحتوي على مجموعة من الأعداد الموجبة سبق تخزينها في الحاسب ، ويراد كتابة برنامج لإيجاد أكبر عنصر في كل صف كي يجمعه على أكبر عنصر في كل عمود مناظر لذلك الصف .

أي أن المطلوب إيجاد مجموع :

أكبر عنصر في الصف الأول + أكبر عنصر في العمود الأول .

أكبر عنصر في الصف الثاني + أكبر عنصر في العمود الثاني .

وهكذا .

٣٠- عند ضرب مصفوفة A تحتوي على M من المصفوف N من الأعمدة في مصفوفة خطية X تحتوي على N من القيم ، فإن ذلك يتم بضرب عناصر كل صف من المصفوفة A في عناصر المصفوفة الخطية وجمع ناتج حواصل الضرب . ولذا فإن النتيجة ستكون مصفوفة خطية أخرى ولتكن Y تحتوي على M من القيم حيث :

$$Y_i = \sum_{j=1}^N a_{ij} * x_j$$

أكتب برنامج يحقق ذلك .

٣١- في التمرين السابق بإفترض أن  $X$  هي مصفوفة أخرى عدد صفوفها  $N$  وعدد أعمدها  $L$  ، فإن عملية ضرب المصفوفتين تم بضرب كل صف من المصفوفة  $A$  في الأعمدة المختلفة للمصفوفة  $X$  حسب الطريقة المبينة في التمرين السابق . وبالتالي فإن مصفوفة حاصل الضرب  $Z$  ( مثلا ) ستحتوي على  $M$  من الصفوف ،  $L$  من الأعمدة ويكون العنصر العام  $Z_{ijk}$  فيها هو :

$$Z_{ik} = \sum_{j=1}^M \sum_{k=1}^L \sum_{j=1}^N a_{ij} * x_{jk}$$

٣٢- أكتب برنامج لإيجاد مقلوب Transpose مصفوفة عدد صفوفها  $M$  وعدد أعمدها  $N$  . ( يمكن الحصول على مقلوب مصفوفة بجعل عناصر الصف  $i$  في المصفوفة الأصلية تمثل عناصر العمود رقم  $i$  في مقلوبها ) .

٣٣- أكتب برنامج لحساب عدد الأرقام الفردية وعدد الأرقام الزوجية المحصورة بين ١، ٩٩٩ .  
٣٤- في برنامج ما يراد طباعة ١٠٠٠ سطر من النتائج . أكتب التعليمات اللازمة لطباعة تلك الأسطر بحيث يكون هناك سطرأ خالياً بعد طباعة كل خمسة أسطر .

٣٥- أكتب برنامج لقراءة أي ثلاث قيم تمثل ثلاث أطوال لبيان ان كانت تلك الأطوال تمثل مثلثاً أم لا . وفي حالة ما اذا كانت تلك الأطوال تمثل مثلثاً ، اختبر ما اذا كان مثلثاً قائم الزاوية وذلك بطبق قيم الأطوال الثلاثة ووضع إشارة أو علامة أمام كل ثلاث قيم تمثل مثلثاً قائم الزاوية .

٣٦- نصف قطر الدائرة التي يمكن رسمها داخل مثلث يعطي من القانون :

$$R = \sqrt{\frac{S(S-a)(S-b)(S-c)}{S}} ; S = \frac{1}{2} (a + b + c)$$

حيث  $a, b, c$  هي أطوال أضلاع المثلث . استخدم التمرين السابق كبرنامج جزئي فرعي لاختبار ما اذا كانت الأطوال  $a, b, c$  تكون مثلثاً أم لا .

٣٧- زمن الذبذبة بالتواتر  $T$  لنبول طوله  $L$  بوصة يعطي من العلاقة

$$T = \pi \sqrt{\frac{L}{g}} \text{ حيث } g$$

تمثل عجلة الجاذبية الأرضية بالبوصة لكل ثانية مربعة . أكتب برنامج لحساب زمن الذبذبة لنبول يتغير طوله من ٦ بوصات الى ٧٢ بوصة بزيادة قدرها ٦ بوصات ، بافتراض أن عجلة الجاذبية الأرضية هي ٨٨ . ٠ ر ٣٨٦ بوصة / ث<sup>٢</sup> .

٣٨- يتم تحويل الدرجة  $X$  بالنظام «الستيني» الى الدرجة  $XR$  بالنظام «الدائري» باستخدام القانون :

$$XR = \frac{X \cdot PI}{180}$$

حيث  $PI$  هي النسبة التقريبية 3.14159

أكتب برنامج لقراءة قيم  $X$  بين ٠٩٠ ، ١٨٠ وكتابة قيم  $XR$  المناظر بحيث :

- تزيد كل درجة عن سابقتها بمقدار ١٠

- يتم استخراج النتائج على الصورة التالية :

DEGREES b CONVERSION b TABLE	
$X$	$XR$
xxx	xx.xxxxxx

٣٩- أقصى ارتفاع  $H$  بالأقدام يمكن أن تصل إليها قذيفة نارية تطلق بزاوية  $a$  مع المستوى الأفقي تعطي من العلاقة :

$$H = \frac{V^2 \sin^2 a}{2g}$$

حيث  $v$  هي السرعة الابتدائية للقذيفة محسوبة بالقدم / ث ، كما أن الزاوية  $a$  محسوبة بالتقدير الدائري وتمثل  $g$  عجلة الجاذبية بالقدم / ث<sup>٢</sup> . فإذا كانت السرعة الابتدائية للقذيفة ثابتة وقدرها ١٠٠ قدم / ث . أكتب برنامج لحساب الزاوية  $a$  التي تصل فيها القذيفة الى أقرب ارتفاع قدره ١٠٥ قدما ، علما بأن الزاوية  $a$  يتم قرائتها بالتقدير الستيني وتتنغير من ١٠ درجات الى ٩٠ درجة بزيادة قدرها ٥ درجات .

٤٠- يمكن حساب عزم القصور الذاتي  $m$  لمستطيل أبعاده  $a$  ،  $b$  حول محوره من العلاقة :

$$m = a^3 b^3 / 6 (a^2 + b^2)$$

أكتب برنامج لقراءة قيم المتغيرات  $a$  ،  $b$  وحساب القيمة  $m$  المناظر لتلك القيم .

٤١- يمكن حساب جملة مبلغ أسامه P وتم استشاره n من السنوات بعائد سنوي معدل i (معبرا عنه في صورة رقم حقيقي) ، وبحيث يحسب العائد m من المرات كل عام ، من العلاقة :

$$A = P \left( 1 + \frac{i}{m} \right)^{n.m}$$

فاذا كان هناك ١٠٠٠ عميل يريدون معرفة جملة مبالغهم بعد عشر سنوات (n=10) وبمعدل عائد ٦٪ (i=0.06) ويتم حساب العائد كل ١ سنة (m=4) . أكتب برنامج لقراءة رقم كل عميل (xxxxxx) وأساس مبلغه المستمر (xxxxxxxx.xx) وطباعة رقم العميل وأساس مبلغه المستمر وجملة ذلك المبلغ في نهاية العشر سنوات . رتب طباعة البيانات السابقة بكتابتها تحت عنوان مناسب .

٤٢- اللوغاريتم الطبيعي لأي عدد بين صفر ، ٤ يمكن الحصول عليه بالتقريب من المتسلسلة اللانهاية .

$$\ln(X) \approx 2 \left[ \frac{x-1}{x+1} + \frac{1}{3} \left( \frac{x-1}{x+1} \right)^3 + \frac{1}{5} \left( \frac{x-1}{x+1} \right)^5 + \dots + \frac{1}{n} \left( \frac{x-1}{x+1} \right)^n \right]$$

حيث n عدد فردي :

أكتب برنامج جزئي فرعي لحساب بعض القيم في هذا المدى (٠ - ٤) ، وأوجد عدد الحدود التي يجب إستخدامها في المتسلسلة السابقة كي يكون الخطأ المطلق بين القيمة المحسوبة بالمتسلسلة ونظيرها المحسوبة عن طريق الدوال سابقة التخزين أقل من أو يساوي ١٠<sup>-٥</sup> .

٤٣- يمكن حساب قيمة e<sup>x</sup> بطريقة تقريبية وتكرارية باستخدام العلاقة :

$$e^x = 1 + \sum_{i=1}^N \frac{x^i}{i!}$$

أكتب برنامج جزئي فرعي لحساب تلك الدالة مستخدما قيما مختلفة للمتغير x . في كل حالة أوجد عدد الحدود التي يجب جمعها كي يكون الخطأ المطلق بين القيمة المحسوبة بالصيغة السابقة ونظيرها المحسوبة عن طريق الدوال سابقة التخزين أقل من أو يساوي ١٠<sup>-٥</sup> .

٤٤- صيغة استيرلنج التقريبية لحساب مضروب عدد ما N ، (N!) يعطى من العلاقة :

$$N! \approx 2 \pi \cdot n \cdot (n^n e^{-n})$$

أكتب برنامج جزئي لدالة لحساب تلك الصيغة لاعداد صحيحة موجبة بين ٢ ، ٩ .

٤٥- في الثمرين رقم ١ ، أكتب كل جزء في صورة برنامج جزئي لدالة .



٤٦- في الممرين رقم ٢٦ ، أكتب برنامج لقراءة المصفوفة الخطية X ، ثم أكتب برنامج جزئي فرعي لاستخراج أكبر الأعداد وكذلك أصغرها وموقعهما في تلك المصفوفة .

٤٧- في الممرين رقم ٢٨ ، أكتب برنامج لقراءة المصفوفة  $A_{MXN}$  . أكتب برنامج جزئي فرعي لاستخراج أكبر وأصغر عنصرين في تلك المصفوفة وموقعهما في المصفوفة .

٤٨- في الممرين رقم ٣٩ ، أعد كتابة البرنامج بحيث يتم حساب قيمة H عن طريق برنامج جزئي لدالة .

٤٩- كُتب برنامج جزئي لدالة لحساب العلاقة التالية :

$$1) S = A \cos x - B \cos y + c - \cos (x-y)$$

حيث :

$$A = \tan^{-1} y, B = \sin^{-1} \left( \frac{x}{y} \right), C = A.B / (A+B)$$

$$2) V = \frac{1}{4} \pi d^2 \sqrt{\left( \frac{gd}{4n} \right) \frac{1}{wa_1} (a_1^2 - a^2)}$$

٥٠- المصفوفة الخطية LIST تحتوي على ٥٠٠ اسمًا مختلفًا وكل اسم مكون من ٢٠ حرفًا ورقمًا . أكتب برنامج جزئي فرعي يُعطى له اسم ما يتم قراءته عن طريق البرنامج الرئيسي كي يقوم البرنامج الجزئي الفرعي بمقارنة هذا الاسم بالاسماء الموجودة داخل المصفوفة الخطية . وتكون نتيجة البحث في صورة متغير يأخذ القيمة YES في حالة العثور على الاسم داخل المصفوفة ، والقيمة NO في حالة عدم العثور عليه .



## قاموس المصطلحات



## قاموس المصطلحات

أرقام الصفحات

## « أ »

١٦٤،١٦٢	I. B. M. (International Business Machines) .....	أ. ب. م.
٣١	Continuation Field .....	اتصال (مجال)
٧	Voice Communication .....	اتصال صوتي
٢١	Output .....	إخراج بيانات
٤٩	.....	إدخال وإخراج البيانات (انظر تعليمات)
٢١	Input .....	إدخال بيانات
٧١	IF .....	إذا (تعليلة)
٣٣	INTEGER NUMBERS .....	أرقام صحيحة
٣٣	Floating-point numbers .....	أرقام ذات علامة عشرية متحركة
٢٣، ٢٨	REAL .....	أرقام ذات فواصل عشرية
٥٧	Alphanumeric .....	أرقام وحروف
٥٩	Exponent .....	أس
٨	.....	استخراج النتائج (انظر وحدة)
١٠٥	.....	استمرار أو مواصلة (انظر تعليلة)
١٥	.....	أسية (انظر متسلسلة)
٢٥	ADD .....	أضف
٥	Numbers .....	أعداد
٢٥	Read .....	اقرأ
٧٥	Less Than .....	أقل من
٧٥	Less or Equal .....	أقل من أو يساوي
٧٥	Greater Than .....	أكبر من
٧٥	Greater or Equal .....	أكبر من أو يساوي
٥٤، ٢٦، ٢٥	WRITE .....	أكتب
١٧٦، ١٧٥	.....	الاستدعاء (انظر تعليلة)
٣٩	.....	التصيين (انظر جملة)
١٨١	.....	التصميم (انظر تعليلة)
١٨٧	.....	التكافؤ (انظر تعليلة)
١٢	Algorithm .....	الخوارزم

١٦٤	الجيب (انظر دالة)
١	Analytical Engine ..... آلة تحليلية
١٩	Calculator ..... آلة حاسبة
١٦٤	الظل (انظر دالة)
١٩	آلي (حاسب) (انظر حاسب)
٧٩، ٧١	انتقال (انظر تعليمات)
١٤٦	انفاذ (انظر مصفوفات)
٢١	انسبائي (انظر مخطط)
٥	Statements or Instructions ..... أوامر (تعليمات)
٢٨	انقاف (انظر توقف)

---

## «ب»

٢٥، ٢١	بداية أو نهاية (انظر نقطة)
٢، ١٣	Program ..... برنامج
٢٩	Writing Program ..... - (كتابة)
٢٩	Submission ..... - (تقديم)
٢٩	Execution ..... - (تنفيذ)
١٦١	Sub-program ..... برنامج جزئي
١٧٣	Sub-routine ..... فرعي
٥	بطاقات مثقبة (انظر وحدة)
٧	مخططة (انظر وحدة)

---

## «ت»

١٥	تحكم ورقابة (انظر وحدة)
١	Analytical Engine ..... تحليلية (آلة)
٣١، ٢٩	Label Field ..... تعريف (مجال)
٣١، ٣٥	Statement Field ..... تعليمات (مجال)
٧٩، ٧١	Branch & Transfer Statements ..... تعليمات الانتقال
٤٥	Order ..... ترتيب
٦٨	Compilation ..... ترجمة وتحويل البرنامج
٥	Statements or Instructions ..... تعليمات (أوامر)

٤٩	Input/Output	- ادخال واخراج البيانات
٦٧	Control statements	- التحكم
٨		- تخزين (انظر وحدة)
٢٩	Submission	- تقديم البرنامج
١٩	Repetition	- تكرار
	Statement	- تعليمة
٧٥	Executable	- تنفيذية
١٠٠,٩٦	DO	- حلقة التنفيذ المتكرر
١٠٥	Dummy	- خادعة أو زائفة
١٧٥	CALL	- الاستدعاء
١٠٥	CONTINUE	- الاستمرار أو المواصلة
١٨١	COMMON	- تعليمة التعميم
١٨٧	EQUIVALENCE	- التكافؤ
١٦٨	RETURN	- الرجوع أو العودة
٢٥	STOP	- توقف

## «ث»

٤٩	Constant	- ثابت
----	----------	--------

## «ج»

١٦١	Statement	- جزئي (انظر برنامج)
	Statement	- جملة أو تعليمة
٦٧		- التحكم (انظر تعليمات)
٥٥,٥٠	FORMAT	- الشارحة
٥٤	WRITE	- الطباعة
٣٩	Assignment	- التعيين
٥٠	READ	- القراءة
١٧٣,١٣٥	DIMENSION	- تحديد البعد
٣٤	Declaration	- مينة أو توضيحية
١٦٤		- جميع الهمام (انظر دالة)

## «ح»

١	Calculi .....	حسابات .....
١	Electronic Digital Computers .....	- الحاسوبية عديدة .....
١٩	Computer .....	حاسب آلي .....
١٩	.....	حاسبة (انظر آلة) .....
٧	.....	حبر مخفط (انظر وحدة) .....
٩٦	Round off .....	حذف التقريب الدائري .....
٥٦	.....	حرفية (انظر واصفات) .....
٥	Letters .....	حروف .....
٥٧	Alphameric .....	حروف وأرقام .....
٢٢	.....	حسابية (انظر عملية) .....
٣٤,٣٣	REAL .....	حقيقية أو عشرية .....
٣٤,٣٣	REAL NUMBERS .....	(أرقام) - .....
٩٩	Loop .....	حلقة .....
١٠٠,٩٦	DO .....	حلقة تنفيذ متكررة .....
١٠٩	Nested DO Loops .....	حلقات التنفيذ المتداخلة .....
١٤٤	Implied DO list .....	حلقات مباشرة .....

## «خ»

٥٧,٣٠	Blank .....	خالية .....
٦٨,١٥	Error .....	خطأ .....
٩٦	Truncation .....	- الانهاء المبكر .....
٩٦	Round off .....	- حذف التقريب الدائري .....
١٠٠	Increment or Step .....	خطوة .....
٣١	Cell .....	خلية .....

## «د»

Function	دالة
١٦٢ SQure Root	الجذر التربيعي
١٦٣ ABSolute value	القيمة المطلقة



١٦٣	EXponential ..	أسية
١٦٤	SINe ..	الجيب
١٦٤	TANgent ..	الظل
١٦٤	COSine ..	جيب الممام
١٦٢	Library Function ..	سابقة التخزين
١٦٦	Arithmetic Statement Fun. ..	دالة ذات تعبير رياضي
١٦٧	Function Subprogram ..	في صورة برنامج جزئي
١٦٣	LOGarithm ..	لوغاريم
٢٩	Logon ..	دخول في النظام
١٣١	Index or Subscript ..	دليل عددي
٢	Integrated Circuits ..	دوائر متكاملة

## « ز »

١٠	.....	رقابة وتحكم (انظر وحدة)
١	Digital ..	رقمي (حاسب)
٣٠	Character ..	رمز
٥٥	Carriage Control Charact. ....	التحكم الحركي

## « س »

٣٠	Continuation Line ..	سطر (متواصل)
٣٠	Comment Line ..	(الملاحظات)

## « ش »

١	Charles Babbage ..	شارل باباج
٧	.....	شاشات (انظر وحدة)
٥٥	.....	شارحة (انظر جملة)
١٠	Operation code ..	شفرة العملية
٢٤	TYPE List ..	شكل عام

## «ص»

٢٤٠٣٣	INTEGER	.....	صحیح
٢٣	Integer numbers	.....	(أرقام) صحیحہ
٢٨		.....	صفحة (انظر هيئة)

---

## «ط»

٥٤٠٨		.....	طابعات (النظر وحدة)
٢٩	Writing program	.....	طباعة (البرنامج)
٢	System	.....	طرز (نظام)

---

## «ع»

٥	Special Characters	.....	علامات خاصة
	Operation	.....	عملية
٢٢	Arithmetic	.....	— حسابية
٧٢٠٣٦	Mixed Mode	.....	— مزدوجة
٢٢	Logical	.....	— منطقية
١٦٨		.....	عودة أو رجوع (انظر تعلية)

---

## «غ»

٧٩	Un conditional	.....	غير مشروطة
----	----------------	-------	------------

---

## «ف»

١٠٠٤٧٧٠٥١	Comma	.....	فاصلة
i	FORTTRAN (FORMula TRANslation)	.....	فورتران

---

## «ق»

٥	.....	قراءة البطاقات (انظر وحدة)
٦	.....	- الشرائط الورقية المثقبة (انظر وحدة)
٦	.....	- الشرائط أو الأقراص المنقطة (انظر وحدة)
٥٦	Quote Mark .....	قوس صغر
٣٣	Data .....	قيم المتغيرات
٣٣	Data types .....	(أنواع) -
	Value .....	قيمة
٩٩	Initial or Starting .....	- ابتدائية
٩٩	Final value .....	- نهائية

## «ك»

٧	.....	كاسيتات مغلطة (انظر وحدة)
---	-------	---------------------------

## «ل»

٧٥	Not Equal .....	لا يساوي
----	-----------------	----------

## «م»

٣٤	.....	مبينة (توضيحية) (انظر جملة)
١٥	Power series .....	متسلسلة أسية
	Field .....	مجال
٣١،٣٠	Statement .....	- التعليمات
٣١	Continuation .....	- اتصال
٣١،٢٩	Label .....	- التمرير
٤٩،٣٩	Variable .....	متغير
٣٤	REAL .....	- حقيقي
٣٤	INTEGER .....	- صحيح
٢١	FLOWCHART .....	مخطط التدفق الانسيابي

٦	Tape recorders .....	مسجلات
٨٣	Computed .....	مشروطة أو محسوبة
١٦٢	Compiler .....	ممثل أو مجمع
١٢٧	Arrays .....	مصفوفات
١٤٦	Array transmission .....	(انقاز) -
٤٩, ٣٩	Expression .....	مقدار جبري أو عملية حسابية
٤٩	Constant .....	- ثابت
٢٢	.....	منطقية (انظر عملية)

## « ن »

٥	Electric pulses .....	نبضات كهربائية
٢	System .....	نظام (طرز)
٢	Integrated circuits .....	- دوائر متكاملة
	Point .....	نقطة
٢٥, ٢١	Terminal .....	- بداية أو نهاية
١٧٣, ١٦٦, ٢٨	END .....	نهاية (للبرنامج)
٢٣	Type .....	نوع

## « هـ »

١	Herman Hollerith .....	هيرمان هولريث
٢٨	FORMAT .....	هيئة أو صفة

## « و »

٥١	Data descriptor .....	واصف بيانات
٥٧	Literal descriptors .....	واصفات حرفية
	Unit .....	وحدة
٥	Input data unit .....	- ادخال التعليمات والبيانات
٨	Output .....	- استخراج النتائج
٥٤, ٥٠, ٤٦	Teletype devices .....	- طرفية

٦	Terminal	طرفية متصلة بالحاسب
٨	Storage or Memory	التخزين
٧	Video devices or Cathode Ray Tube Screens	الشاشات
٥٤٨	Line printers	الطابعات
٧	Magnetic Cassets	الكاسيتات المغنطية
٧	Magnetic Cards	بطاقات ممغنطة
٧	Magnetic Ink	حبر ممغنط
١٠	Control	رقابة أو تحكم
٥٠٥	Card reader	قراءة البطاقات
٦	Punched paper tape reader	قراءة الشرائط الورقية المثقبة
٥٤٥٠٦	Magnetic tape or Disc read	قراءة الشرائط أو الأقراص المغنطية

## « ي »

٧٥	EQual	يساوي
----	-------	-------



« المراجع »





- 1 - Frank L. Friedman and Elliot B. Koffman : Problem Solving and Structured Programming in Fortran, 2nd ed., Addison- Wesley, 1981.
- 2 - Hossam El-Bebrawi : Computer Glossary, Studies in Automatic Al-Maaref Estab., 1975.
- 3 - Meissner / Organick : Fortran 77 Featuring Structured Programming, Addison-Wesely, 1980.
- 4 - Rich Didday and Rex Page : Fortran for Humans, 3rd ed., West, 1981.
- 5 - Robert H. Hammond; William B. Rogers and Byard Houck : Introduction to Fortran IV, 2nd ed., McGraw Hill, 1978.
- 6 - William F. Schallert and Carol Reedy Clark : Programming in Fortran, Addison Wesley, 1979.
- 7 - William S. Davis : Fortran Getting Standard, Addison- Wesley, 1981.







مطابع جامعة الملك عبد العزيز